REPORT NO. GDC DDF66-007

AUTOMATIC MALFUNCTION ANALYSIS BY DISCRETE NETWORK SIMULATION

APPENDIX C
PROGRAMMERS REFERENCE MANUAL
AUTOMATIC MALFUNCTION ANALYSIS PROGRAMS

GENERAL DYNAMICS
Convair Division

REPORT NO. GDC DDF66-007

AUTOMATIC MALFUNCTION ANALYSIS BY DISCRETE NETWORK SIMULATION

APPENDIX C
PROGRAMMERS REFERENCE MANUAL
AUTOMATIC MALFUNCTION ANALYSIS PROGRAMS

OCTOBER 1966

Submitted to
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GEORGE C. MARSHALL SPACE FLIGHT CENTER
HUNTSVILLE, ALABAMA
under
CONTRACT NAS8 – 20016

Prepared by
CONVAIR DIVISION OF GENERAL DYNAMICS
HUNTSVILLE, ALABAMA

TABLE OF CONTENTS

PART ONE

AUTOMATIC MALFUNCTION ANALYSIS PROGRAM

PART TWO

AUTOMATIC MALFUNCTION ANALYSIS EDITOR PROGRAM

AUTOMATIC MALFUNCTION ANALYSIS PROGRAM (#3998)

AUTHOR

D. R. Diaddigo
General Dynamics/Convair
Scientific Programming and Analysis (595-0)
8 August 1966

PURPOSE

For each point of scan in the test procedure for the S-1C checkout system, the DNS simulator produces a list of the component states for proper functioning of the system. The Automatic Malfunction Analysis Program then considers, on the assumption the modelled Discrete Inputs are monitored, what modelled components failures in the system could cause various DI's or combination of DI's to fail. It produces a list of effect - cause relationships for the AMA-Editor Program (4012) which produces the search tapes for the RCA110.

RESTRICTIONS

(1) Program must run on 7090/94 with IBJOB systems capability

(2) In addition to system input and output, four additional magnetic tape units are required.

STORAGE

Program/Sub-program	Function
A. FORTRAN IV	
1. Amaz	1. Driver
2. Analyz	2. (a) Determine variables that can affect the value of equation (b) Identify and flag conditionals (c) Identify and flag reverse legs

3. Conclx	3. Auxiliary conditional analysis routine - solution of conditional string of variables
4. Condit	4. Auxiliary conditional analysis routine - creation of conditional variable and branch flags.
5. Data	5. Block data - for tapes and storage limits
6. Docasz	6. Main logical control over conditional and unconditional procedure for each list
7. Enctex	7. Auxiliary routine to carry effects into higher common branch points.
8. Evaluz	8. Evaluation of equation according to list input to verify consistency
9. Expcx	9. Auxiliary conditional analysis routine - rearrangement of conditional information to aid in solution
10. Fcasez	10. Set up tables according to current list for solution and tape information for editor
11. Fcfcz	11. Output routine (a) Effect - cause BCD print (b) Effect - cause records for AMA Editor Program
12. Lctz	12. Auxiliary routine for page titling and line control
13. Negstx	13. Auxiliary analysis routine to check for stopping on reverse leg processing
14. Packz	14. Auxiliary routine to control causes sequencing through model equations
15. Presox	15. Main logic driver for solution of conditional and creation of conditional solutions table
16. Setcax	16. Initialization of flags and tables prior to processing of list
17. Setupz	17. Initial table read in and set up

18. Soltrx — 18. Storage of conditional solutions table

19. Subsol — 19. Substitution of conditional solution for encountered conditional variable

B. IBMAP:

1. Atapez — 1. Reading of Simulation List input tape

2. Convrx — 2. Binary to BCD translation

3. Edstat — 3. Packing of DI states for list for AMA Editor

4. Effrec — 4. Reduction of encountered matrix for effects record for AMA Editor

5. Equexp — 5. Expansion of equation bit coding for processing

6. Fnrins — 6. Insertion of various analysis flags

7. Fnrret — 7. Retrieval of various analysis flags

8. Fnrzer — 8. Deletion of various analysis flags

9. Fud — 9. Identification of special magnetic tapes

10. Konaux — 10. Conditional Analysis flag creation and retrieval

11. Rtvdm — 11. Auxiliary routines for creation of common point levels and bit position identification

12. Simdat — 12. Retrieval of simulation input data flags

13. Stgmnp — 13. Insertion of effected DI bits in common branch points matrix

14. Tranl — 14. Translation from variable code to three letter code

15. Varaux — 15. Unpacking of complete variable status flags

TIMING: 2.5 minutes/list for S-1C model

USE: To set up program for operation 4 magnetic tapes and one specification card are required

A. Tapes

Fortran Logical	System Function	Tape Function
-	A(2)	Simulation Input
8	A(1)	DT&C AMA tape
11	B(1)	AMA Editor tape
9	B(2)	Terminal variable matrix

B. Specifications Card (all values integer)

Field	Columns	Function
1	1-6	Test Control 0 - Any reactions remaining terminates job 1 - Any reactions remaining in list terminates the list 2 - Any reactions remaining inconsistencies are noted, action discontinues on branch, but processing continues in list
2	7-12	Output Option 0 - BCD print of effect, causes and AMA Editor tapes 1 - BCD print only 2 - AMA Editor tape only
3	13-18	The first N variables are to be treated as common branch points
4	19-24	The first M variables are the active variables in the model

METHOD:

A. First procedure executed by program is set up of basic tables.

(1) The input tape from DT&C program (3843A) is read for the coded equations table and the function reference table.

(2) The input from preprocessor - editor program (4019) is read and the static common point level of each of the common branch points is computed and inserted into the function reference table.

(3) Limits for:

a) Equation table
b) Function reference table,
c) Encountered common branch point matrix, and
d) Working area for analysis are defined

(4) Flags on the DI's are set

B. Each list encountered from the simulator input tape institutes the following procedure.

(1) All flags (state, encountered, processed, etc.) are cleared from function reference table.

(2) The input state list is processed and the states are inserted into the function reference table. The packed state list for the DI's are transmitted to the AMA Editor.

(3) An initial pass over the function reference table is made to identify conditional functions and find solutions for them. A section of the working area is then assigned to hold the conditional function solution and appropriate flags are set in the function reference table to identify the conditional and to indicate whether it has a solution.

(4) For each common point level two passes are made over the function reference table (common branch point section) and for each common branch at this level the analysis procedure is initiated.

a) 1. In the first pass the actual common branch relationship is determined.

2. In the second pass the failure components are investigated.

b) The first variable encountered is the common branch itself and the terminal DI's that it will affect are either the DI itself, if

the process is at level one, or a combination of DI's, stored by bits in the Encountered Common Branch Matrix, if the process is at a higher level.

c) The equation for the variables under consideration is expanded producing a set of variables, whose equations will also be expanded. The set of variables consists of those in the equation whose assumption of the opposite value than the one they hold will change the value of the equation in which they occur. Each variable is investigated for following conditions.

1. Common branch level

   a) If higher branch level, the processing of the variable is stopped, until its level is processed, and it is flagged as encountered with the terminals through which it has been encountered inserted into its position in the Encountered Matrix.

   b) If of the same level, the variable is investigated further.

2. Negation Stoppage

   If the negation flag is set the variable is the reverse leg opposite to the direction of Analysis, whose failure would be a contradiction. The variable is then deleted from the set.

3. Failure Candidacy

   Dependent on the value of the variable the variable is inserted into the causes list.

4. Conditional - if the variable is identified as a conditional, its solution if it exists is retrieved from the solution list and added to the set, and the variable is deleted from the set.

5. Initiator

   If the variable has no further expansion, it is deleted from the set and processing continues with the next variable in the set.

6. Transaction Variable

   If the variable has not been deleted from the set under conditions (1), (2), (4) or (5) its equation is expanded and new variables are added to the list if they can contribute to failure. The variable is then deleted from the set.

At the end of each step the set of variables is repacked so that the next variable to be investigated moves to the top of the set.

When the set is depleted for the initial variable, processing continues with the next variable at the same branch level, after transmitting effect and causes to the AMA Editor.

When variables at this branch level are completed, the program processes the next branch level and upon completion of all branch levels, the list is considered complete and it proceeds to the next case.

d) The common branch points processed are those only actually encountered through lower level branches (c) (1) above. A flag in the function reference table is used for this, and to prevent re-processing variables encountered through various branches, a has been processed flag is also set. (In the prepass, a separate flag is used for the same purpose).

DEFINITIONS:

1. Common Point Level - The preprocessor editor program (4019) in a section of its processing defines the relationship between terminals (Discrete Inputs) and the components in the network. The number of terminals connected to the component is the common point level of the component. It is not necessary to assign common point levels to non-branch variables, since they will have the same level as the branch through which they are encountered.

2. Common Branch Point (Static) - If a variable has a direct effect on variables of lower common point level, then it is a common branch point. The analysis loops are based on static branch point level.

3. Common Branch Point (Dynamic) - In certain model states, a static common branch can become a non-branch point. The case in which this occurs is the modeling for the forward-reverse legs in the system. Dependent upon the current flow direction, the nodes connected by this type leg, can, 1) both be considered common branches with the same effect, 2) both common branches with independent effects, or 3) one node becomes the common branch and the second determines only the connectivity with the rest of the circuit. The criterion for determining the actual common branch status is whether the node is encountered independently of its partner. The prepass, mentioned in the method, is used to determine this and the true effect on the system for these type cases.

FAILURE CANDIDACY

When the value of an equation is zero or one, then only those variables in the equation whose effective value is zero or one can cause failure. But to be entered into the causes list, the component itself must be capable of failure in that state, otherwise it can be used only to establish connectivity.

As an example:

A leg carries power between nodes. If its value is one (current is passing through), it can fail and cut off the flow of current of its own accord. But if no current is passing through then it can fail, assuming power, only if other component, such as a contact, fails in such a way as to permit current through. Thus, in the first instance it will be listed as a possible source of failure, and in the second, it will be used only to lead back to the contact which is the possible source of failure.

EFFECTIVE VALUE

The state of a component may be opposite its effect on the system at a particular time. This is expressed by usage of the /operator. Thus:

A = B * /C where value of C is 0, indicates that its effective value is 1, and C is considered for failure under the conditions that it is 1.

CONDITIONAL AND UNCONDITIONAL FUNCTIONS

If the change of state of individual components in an equation changes the value of the equation then the equation is unconditional , and these variables are entered directly into the set of variables to be processed further.

Examples:

(a) A = B + C + D where the values of A, B, C, D, are zero

Either B, C or D assuming the value of 1 can cause the value of A to become 1, giving three unconditional solutions to the problem.

(b) A = B * C * D where all values are 1. The resultant solutions are as above, B, C and D are the unconditional solutions.

If however, the change of value of an equation depends upon the simultaneous change of state of two or more variables, then the function is conditional. There is a solution to the equation only on the condition that there is a set of variables, anyone of which can cause the simultaneous

failure of the conditional with its failure.

Examples:

(c) A = B * C * D where all values are zero. If B, C and D are in turn the function of the same variable E whose change of state changes their states then E is the solution. Otherwise there is no solution. There of course may be several.

(d) A = B + C * D where the states of all variables are 1. Then the solution to the problem consists of finding those common variables that can cause either B and C to fail or B and D to fail.

A function may have both unconditional and conditional solutions.

Example:

(e) A = B + C * D where all values are 0. The solutions are B and the set of variables that cause both C and D to assume the values of 1 simultaneously.

The program treats the above as conditional with B as one of the solutions.

APPENDIX A: PROGRAM PROCEDURES AND FLOW CHARTS

A. References to data items are made to Appendix B outlining data storage and labeled commons.

B. Subroutines are discussed in alphabetical order with an explanation of their options and techniques and their interrelationship with other programs through labeled commons.

C. Fortran Routines

1. AMAZ - Main driver

   a) Set up tables
   b) Initialize tables for list
   c) Read list and set states
   d) If last list exit
   e) Perform analysis
   f) Continue from b

2. ANALYZ

   CALL ANALYZ (NCPL, ILL)

   Performs analysis on variable (JFUNC) specified in /BRHPRS/

   a. Validates equation value. If value is inconsistent, a message is printed and action taken in accordance with input specification.

   b. Examines each variable on left side of equation for possible malfunction dependent on value of equation. Possible malfunctions are placed in list in /STREF/ under control of NAMLIX of /BRHPRS/ for further processing.

   c. If function is conditional, the conditional variables are flagged with their branch numbers, NNCON of /CONPRS/ indicates position and KONDF is the flag.

   d. Negated variables are flagged in function reference table and their identifications are placed in upper section of /STREF/ under control

of INEG of /BRHPRS/ for flag clearance at end of common branch processing.

e. Common point levels are assigned to variables not placed in the common branch point category during the conditional presolution process by using the common branch point's level through which they were encountered. This is to prevent the attachment of failure causes to the improper level when solution substitution takes place.

f. Analysis contains another error message to indicate if a parenthesis, which is not acceptable in this version of AMA, has occurred in an equation. The run terminates with this error message.

Two dummy variables in calling sequence.

NCPL - 1, routine has been called during conditional presolution (PRESOL). Options taken are to carry down common point levels and ignore inconsistency message.

2, routine has been called from unconditional analysis (DOCASE). Options taken are check for inconsistency and bypass carrying down common point level.

ILL - routine sets ILL to 1, if inconsistency is encountered. Calling program decides on action if control is returned.

3. CONCBX

CALL CONCBR (MST)

Dummy variable MST is set to 1 if all branches of a conditional have terminated in solutions. Otherwise it is set to zero.

Procedure: (1) The conditional identification of the latest variable added to the conditional list

(2) All variables having this identification are separated according to their branch identifications.

(3) The expansion of the conditional list is then solved. Any variable assigned to all branches is a solution. One reference is retained and all others deleted. The variable is not processed further for this conditional.

Routine used only during PRESOL.

4. CONDIX

CALL CONDIT

1. Sets conditional id KONID of /CONPRS/ to 1 for conditional function

2. Carries down the conditional id into each variable assocaited with the conditional function, retaining the branch number.

3. If a conditional has independent term (e.g., A = B * C + D * E, all values zero, there are two terms that can be solved independently for solution) KONID is changed for each.

4. The number of branches is determined, (for zero's conditional, the number of variables in each term are counted, for one's conditional, the highest branch number is the number of branches), and the flag word consisting of KONID, NBRAN, and JFUNC is created for the conditional function.

5. NNCON is positioned, past the flag words, for the first variable in the conditional for further processing.

5. DATA

Block data subroutine

Presents

1. MTITLE to blanks
2. IEND to 18500 (current limit of ISTORE)
3. Tape assingments /IODEFS/

6. DOCASZ

CALL DOCASE

Executes procedures B 3) and B 4) described under Method.

a. Working storage in /STREF/ is defined by position counter NAMLIS.

b. Common point level of operation is defined by ICCPL.

c. The variable under investigation is JFUNC, always a position within /STREF/. The variable status is expanded from the function reference table into /VARSTS/.

d. JFTERM defines the common branch point under analysis. JFTERM is always the variable code number.

e. NAMLIX is a position within /STREF/ used to define the upper limit of possible variable failure associated with the common branch.

f. LSPEC (2) is used to control the output of the effect-causes results.

g. INEG is used to define the beginning position in /STREF/ where the negated variable encountered has been stored. (Table goes to IEND).

7. ENCTEX

CALL ENCTER

If during the processing of a common branch point, a higher common branch is encountered as a failure possibility this routine is entered to transfer terminal effect from the current common branch to the higher encountered common branch.

a. JFTERM defines the branch point which encountered the higher branch.

b. JFUNC defines the branch encountered.

c. Transfer of terminal effects are made from row assigned to JFTERM in encountered Matrix to row assigned to variable defined by JFUNC.

d. If JFTERM is one of the terminals ($\leq$NTERM) its bit configuration is computed and transferred to JFUNC's row.

e. JFUNC is flagged as encountered for processing when DOCASE driver reaches its common branch level.

8. EVALUZ

CALL EVALU

From /EQSKEL/ which contains the expansion of the equation with symbol or variable code per word, EVALU retrieves the current variable values and computes the value of the equation storing it into EQUVAL.

9. EXPCX

CALL EXPCON

Expansion of conditional function variable set for solution by CONCBR routine.

a. Limits of search are defined by NNCON and NAMLIX.

b. Conditional function searched for and number of branches is defined by ID and NBKON of /CONSOL/.

c. Expansion of the conditional goes into /STREF/ starting at NAMLIX + 1.

d. If a variable occurs more than once in the same branch, the later occurrences are erased.

e. The beginning and end positions of the solution Matrix are set into IN and NAL of /CONSOL/.

10. FCASEZ

CALL FCASE (NONE) - Read list and set function reference table for processing of case.

a. Simulation of data is read into IBIOT of /RW/

b. Tests on type of control card (* in first position of first word) against internal data statement for which following actions are taken

1) *TITLE - title (if first) is transmitted to /MTITLE/ for run titling and to AMA Editor. All other titles are ignored.

2) *$$$$$ - transmit 'EOFEOF' to AMA Editor and set NONE to 0 to signal termination of job to driver.

3) *LIST - set flag so that incoming data is treated as variable state data. Transmitted to AMA Editor after block, step, sub-step numbers are converted.

4) *END LIST - creation of profile record for AMA Editor. Erasure of flag for state data.

5) *ACTIONS - set flag for action data.

6) *END ACTIONS - determine if reactions remain and take action as specified by LSPEC (1).

c. List data - if variable occurs as 1 in list set value bit of function reference table to 1 for the variable.

d. Action data - determine if data is reaction or input and maintain counts.

11. FCFCBZ

CALL FCFCB(INAME, IKIND)

INAME $\neq$ 0 store cause INAME into KCFC buffer and increase ICFCC.

INAME = 0 output current buffer under LSPEC (2) option. If option includes print translate buffers to 3 - letter codes before printing.

IKIND = 0 output EFFECT record.

= 1 output CAUSES record.

12. LCTZ

CALL LCT (LENT, LADD, LFA) - Print control

LENT = 1 Increase line count by LADD, if it exceeds maximum (currently 50), reset and print new page title, title only if LFA = 1, subtitle for variable identification if LFA = 2

= 2 Set line count to 0

= 3 Set page count and line count = 0

= 4 Same as 3

13. NEGSTX

CALL NEGSTP (NFLAG, IVAR)

a. Expand equation for IVAR into /STREF/ starting at NAMLIX + 1.

b. If IVAR contains the negation of the JFUNC variable code, set NFLAG = 1.

14. PACKZ

CALL PACKIT

In section of ISTORE of /STREF/ defined by position limiters NAMLIS and NAMLIX, the data is repacked to eliminate cells that have been zeroed out, giving a new value to NAMLIX.

15. PRESOL

CALL PRESOL - Executes section described in B 3) under Method.

a. The storage for conditional solutions, ICONS of /CONCXX/ is set to start at IFREE of /STREF/.

b. The working storage in /STREF/ from IFREE to IEND is divided into two sections.

   1. 1/3 to store conditional solutions, and
   2. last 2/3's for variable lists, negation data, equation and conditional branch expansion.

c. LEVEL is set to control definition of conditional branches for ANALYZ.

d. KONR to indicate conditional identification.

e. NNCON, NAMLIX to define limits of working area where variable set for a particular conditional is stored.

f. ICX - holds position of conditional identification, in variable list of conditional currently under solution.

g. Parameters in /VARSTS/, /BRHPRS/, /REFTAB/ are used in same manner as in DOCASE.

h. ICTSX of /CONCXX/ is set to upper limit of conditional solution storage and NAMLIS for DOCASE will start in the position afterwards.

16. SETCAX

CALL SETCAS

a. Before entry into FCASE, the function reference table is cleared of all flags with exception of encountered flags on terminals, state value on inactive variables, and all common point levels and equation references.

b. The encountered branch point Matrix area is cleared.

17. SETUPZ

CALL SETUP

Procedure outlined under A in Method is exectued.

a. All tape input is into KCFC of /EFFCAU/.

b. Equation table is read into ISTORE of /STREF/ and its position limits IEQT, IEQX are set into /EQUTAB/.

c. Specification card is placed in LSPEC of /RW/.

d. Function reference table is read and /REFTAB/ is set for limits IFNRT, IFNRX.

e. Terminal variable Matrix is read and common level established for common branch points. Maximum common level, ICPMAX, is established.

f. Limit positions are set for terminals (IFTER), common branch points (IFCBT) and active variables (IFNAV) as well as the number of terminals NTERM for /REFTAB/.

g. Function reference table for each variable is expanded into /VARSTS/ and printed.

h. Limit positions ITVDM, ITVDX and row (ITVRW) and column (inwords, ITVCL) are set in /TVMTAB/ for the encountered common branch point Matrix.

i. Encountered flags are set for the terminals.

j. IFREE of /STREF/ is set to indicate the beginning of working storage.

18. SOLTRX

CALL SOLTRF

a. -JFUNC is stored to indicate the variable the conditional solution pertains to.

b. The solution stored between NNCON and NAMLIX is transferred.

c. ICTSX is repositioned to indicate the limit of conditional solution storage.

19. SUBSOX

CALL SUBSOL

a. JFUNC is the solution to be retrieved.

b. Search is conducted between limits ICONS and ICTSX.

c. Solution is transferred into positions starting with NAMLIX + 1 and NAMLIX is reset.

D. Map Routines

1. ATAPEZ

a. CALL BOPEN - open simulator input tape.

b. CALL BOTRED - read simulator record into IBIOT of /RW/.

If tape is redundant, message is printed and job terminated.

2. CONVRX

I = CONVRT (A)

The binary number A is converted to BCD image with leading blanks.

3. EDSTAT

CALL EDSTAT (N)

Operating on function reference table between limits IFNRT and IFTER the list profile on the terminals is created for the AMA Editor into KCFC of /EFFCAU/.

N is set to the number of words in profile.

4. EFFREC

CALL EFFREC (A)

For the row of the encountered branch point Matrix starting at A, determine which of the NTERM terminals are effected by this common branch. Translate the bits into their implied variable code position and store in KCFC and the final count into ICFCC of /EFFCAU/.

5. EQUEXP

CALL EQUEXP (A, N, B)

a. Expands equation whose position reference is A into array B and set total number of operators and variables into N.

b. Each operator is flagged as negative with value between 1 and 7.

c. Each variable code has positive sign.

6. FNRINS

The argument A for all entries indicates a position for the function reference table within /STREF/. 7090 word is S, 1-35.

a. CALL CPLINS (A) Insert common point level from VARCPL of /VARSTS/ into bits (6-17).

b. CALL HBEINS (A) Set has been encountered position, bit 5, to 1.

c. CALL HBPINS (A) Set has been processed position, bit 2, to 1.

d. CALL NEGINS (A) Set negation flag, bit 18, to 1.

e. CALL LEVINS (A) Set conditional branch, or level, bits 3-15 to value of LEVEL. This insertion is not into the function reference table.

f. CALL HBCINS (A) Set has been checked flag, bit 19, to 1.

g. CALL HBKINS (A) Set conditional identification, bit 20, to 1.

h. CALL KSLINS (A) Set conditional solution flag, bit 1, to 1.

7. FNRRET

Arguement A is position within ISTORE

a. I = CPLRET (A) Retrieve common point level as integer B35.

b. I = VALRET (A) Retrieve variable value as integer B35.

c. I = EQURET (A) Retrieve equation reference, bits 21-35, as an index.

d. I = VARRET (A) Retrieve variable code, bits 21-35, as an index.

8. FNRZER

Arguement A is position in ISTORE

a. CALL ZRPV (A) Zero value, processed, conditionality, negation, checked flags (Bit S, 1, 2, 18-20).

b. CALL ZREPV (A) Zero above flags including encountered (Bits S, 1, 2, 5, 18-20).

c. CALL ZREPV (A) Zero all flags except value (Bits 1, 2, 5, 18-20).

d. CALL LEVZER (A) Zero conditionality branch (Bits 3-15).

e. CALL NEGZER (A) Zero negative flag (Bit 18).

9. FUD - tape assignments cards.

10. KONAUX

Argument A is a position within ISTORE.

a. I = KOBRET (A) Retrieve conditional branch(es) number, bits 1-10.

b. CALL KONDEF (A) Expand conditional information into,

NBRAN - branch(es),
KONID - conditional id,
KONR - associated variable.

c. I = JIDRET (A) Conditional level, bits 3-17, retrieval.

d. CALL KONINS (A) Create conditional process word from NBRAN and KONID assuming variable is already in A.

11. RTVDM

a. I = RTVDM (N) translated N into a word and bit position of KCFC and retrieves the bit.

b. J = IBCOV (N, L) translated N into word position J (position start at 0) and store bit configuration into L.

12. SIMDAT

a. I = SCHRET (A) Retrieve first BCD character of IBIOT (1) of /RW/ and insert leading zeroes.

b. I = IXDRET (A) Retrieve address of IBIOT (2) (Index code).

c. I = ITYRET (A) Retrieve data type Bits 3-17 from IBIOT (2).

13. STGMNP

a. CALL MERGI (WORD, A) - Merge WORD into position A of ISTORE.

b. CALL MERSTG (POS 1, POS 2, N) - Merge N words starting at Pos 1 of ISTORE into corresponding number of words starting at POS 2.

14. TRANL

CALL TRANS (N, M)

Translate an index code N into a 3-letter alpha code with trailing blanks. The translated name is stored in M.

15. VARAUX

CALL VARDEF (A)

Unpack the function reference word in position A of ISTORE into /VARSTS/ and CONPRS/.

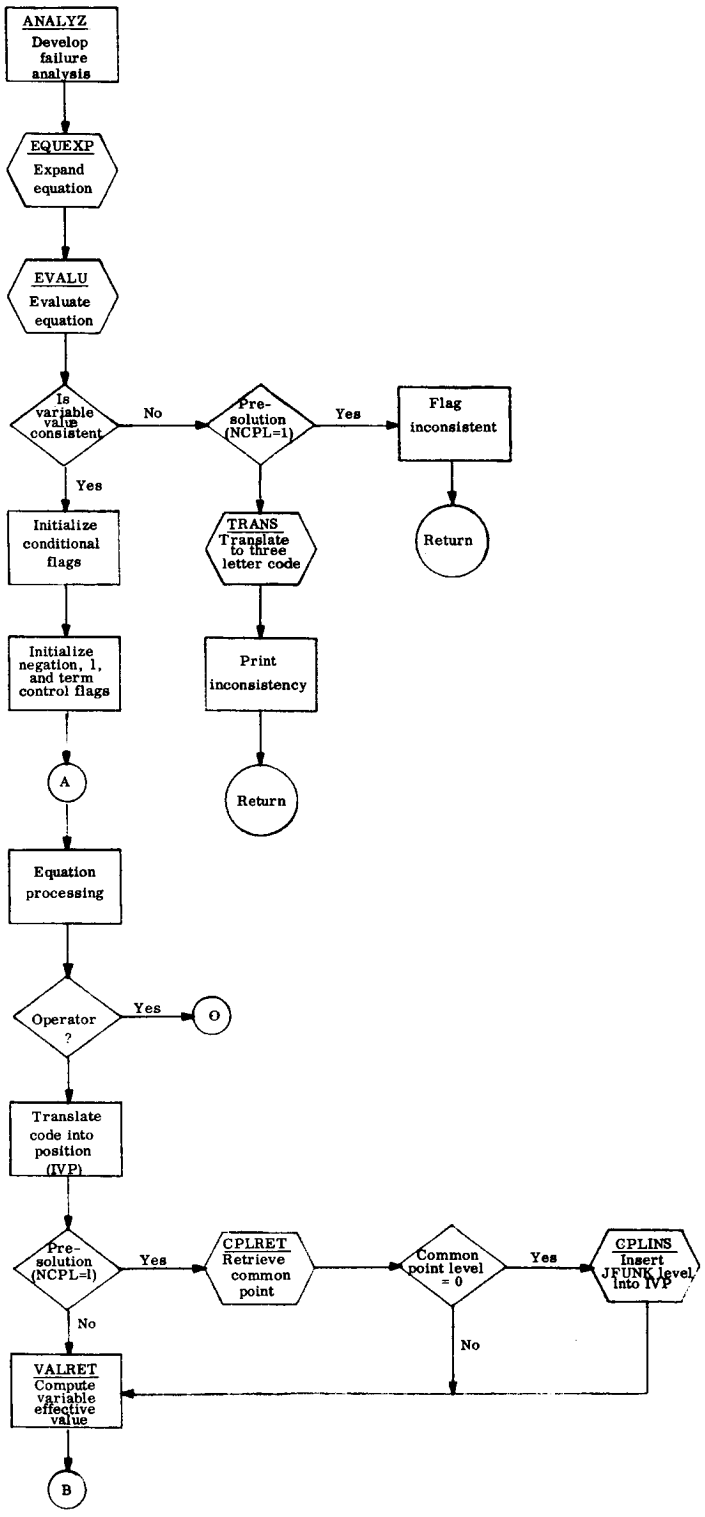


Figure A-1. ANALYZ (1)

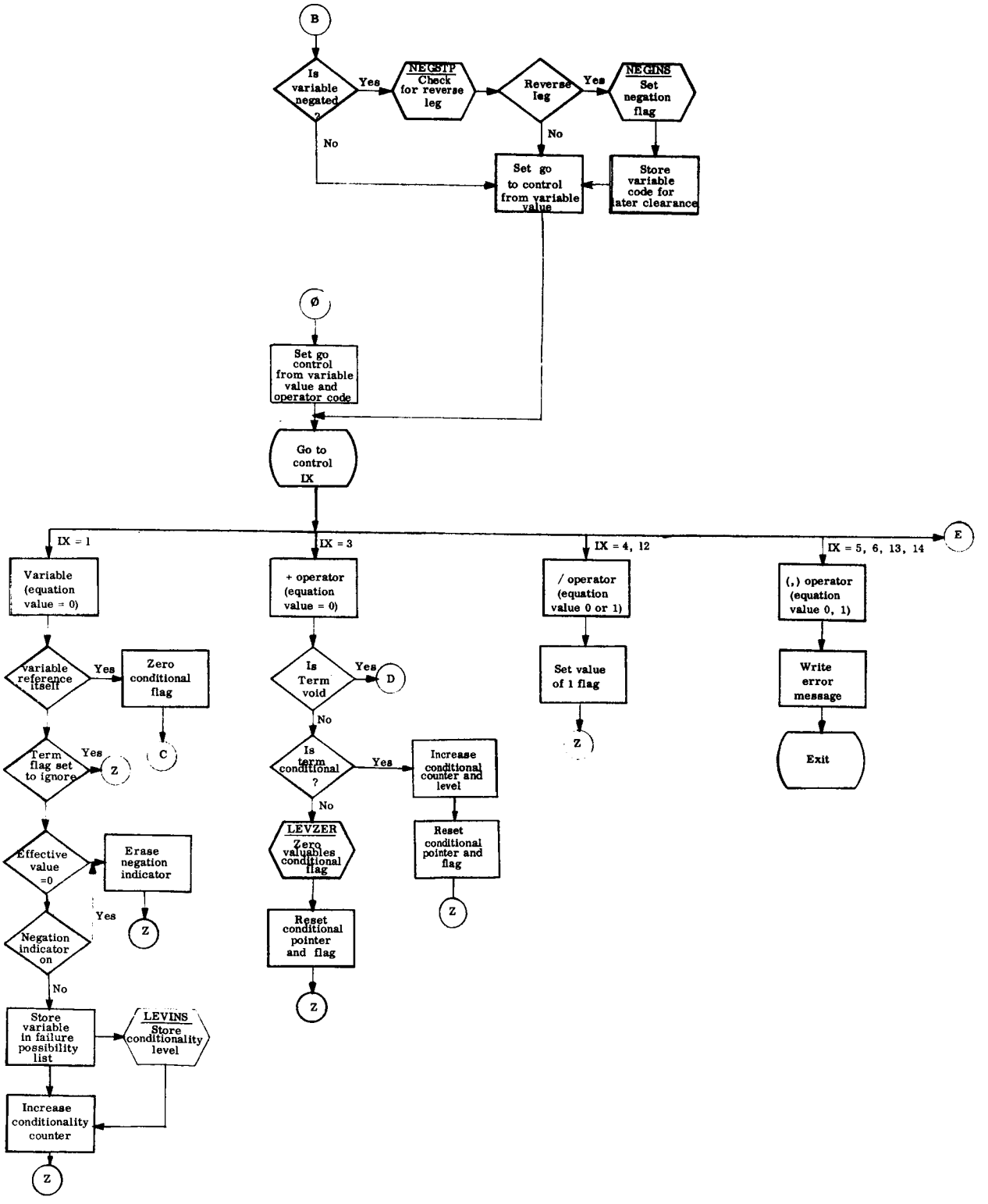


Figure A-2. ANALYZ (2)

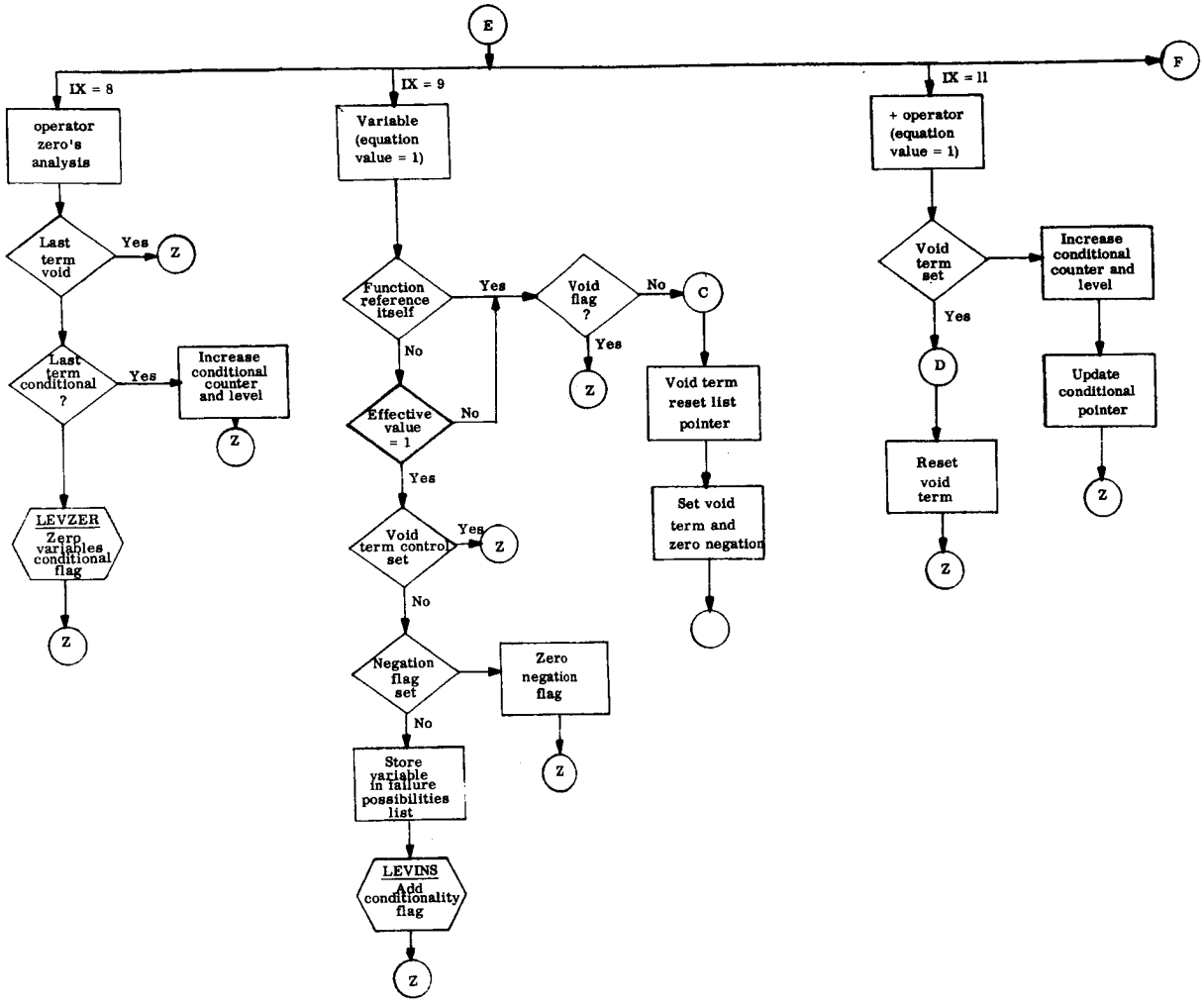


Figure A-3. ANALYZ (3)

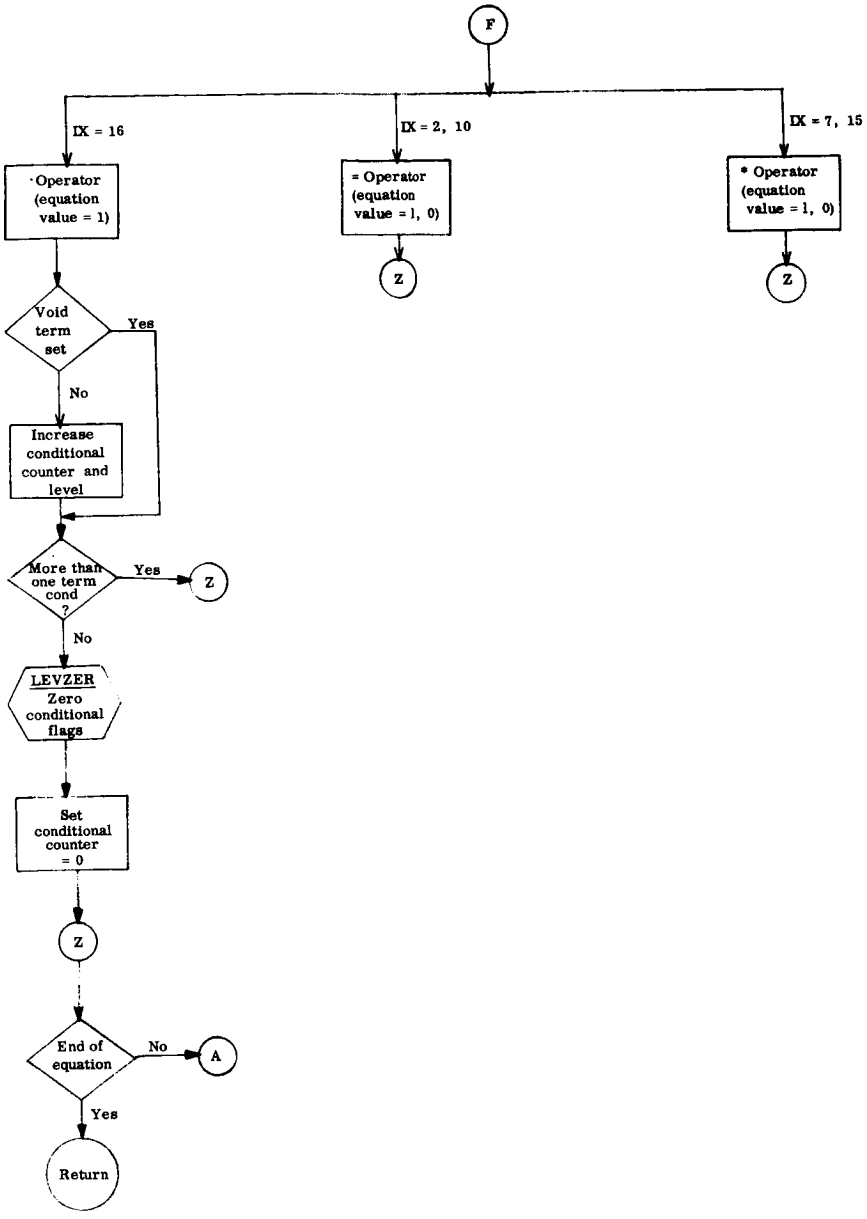


Figure A-4. ANALYZ (4)

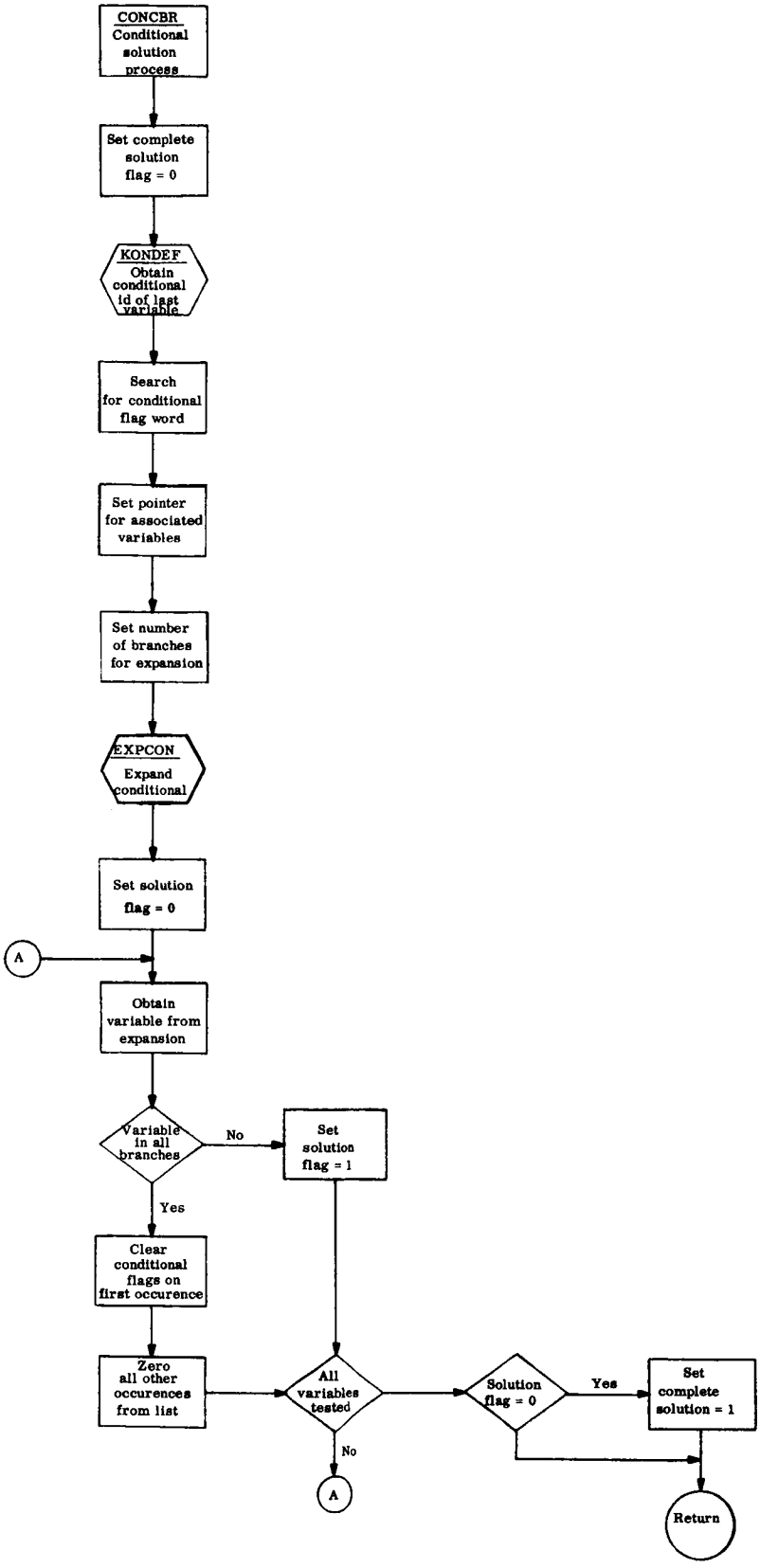


Figure A-5. CONCBX

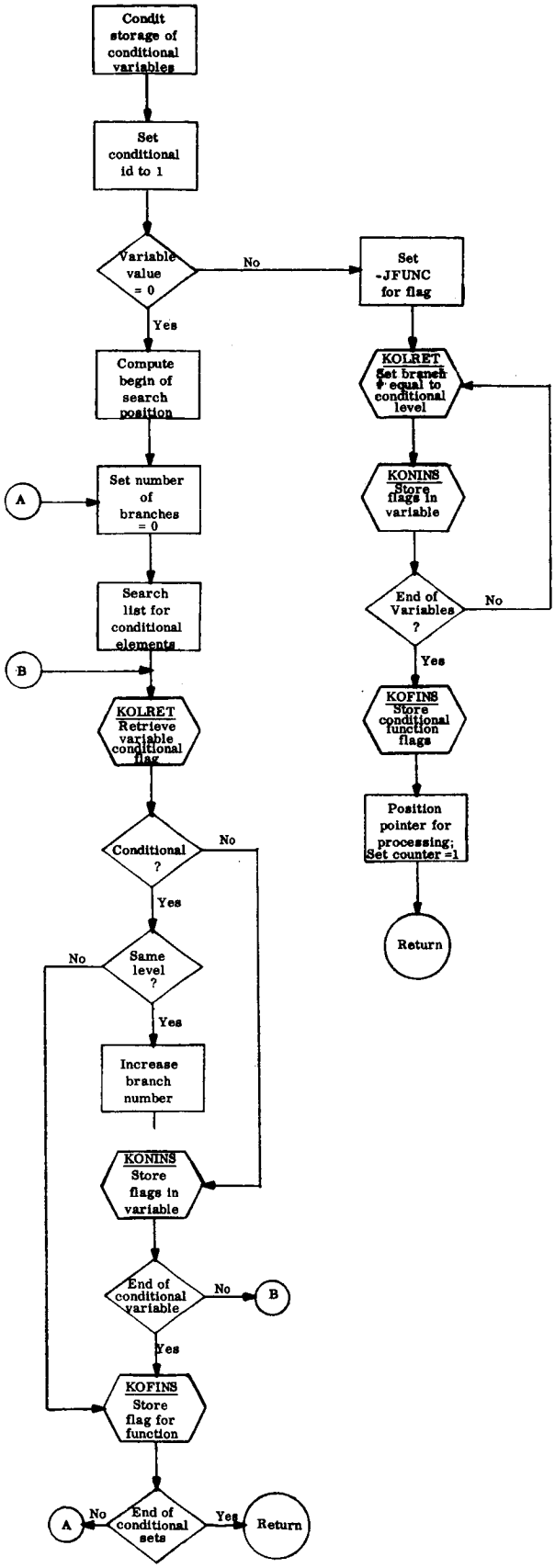


Figure A-6. CONDIX

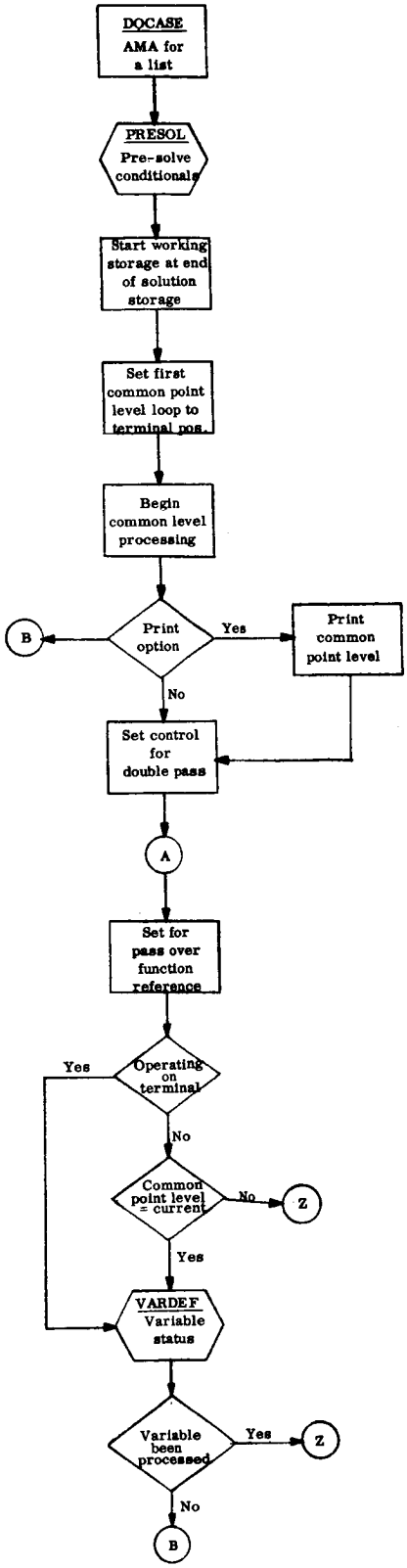


Figure A-7. DOCASE (1)

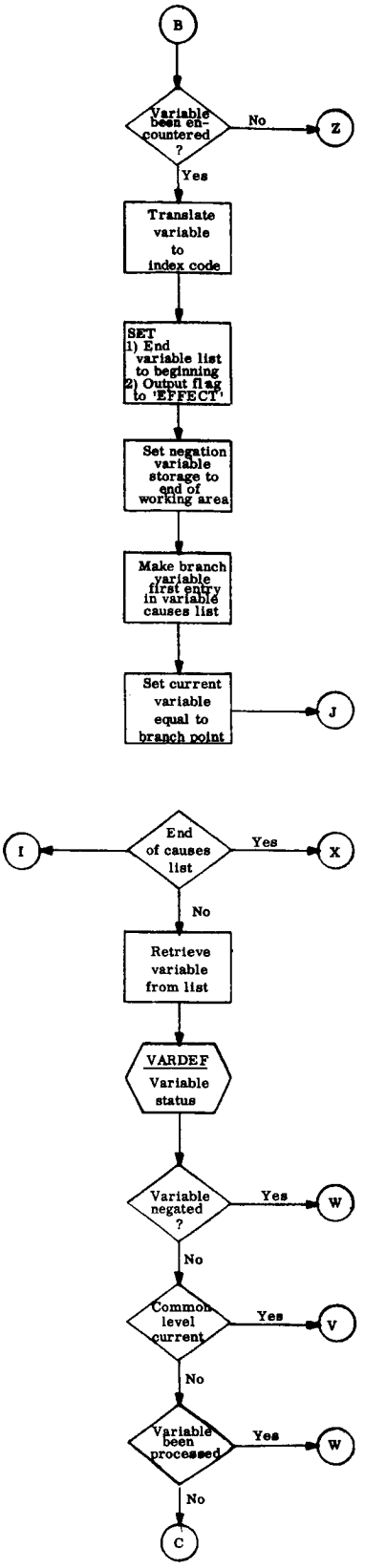


Figure A-8. DOCASE (2)

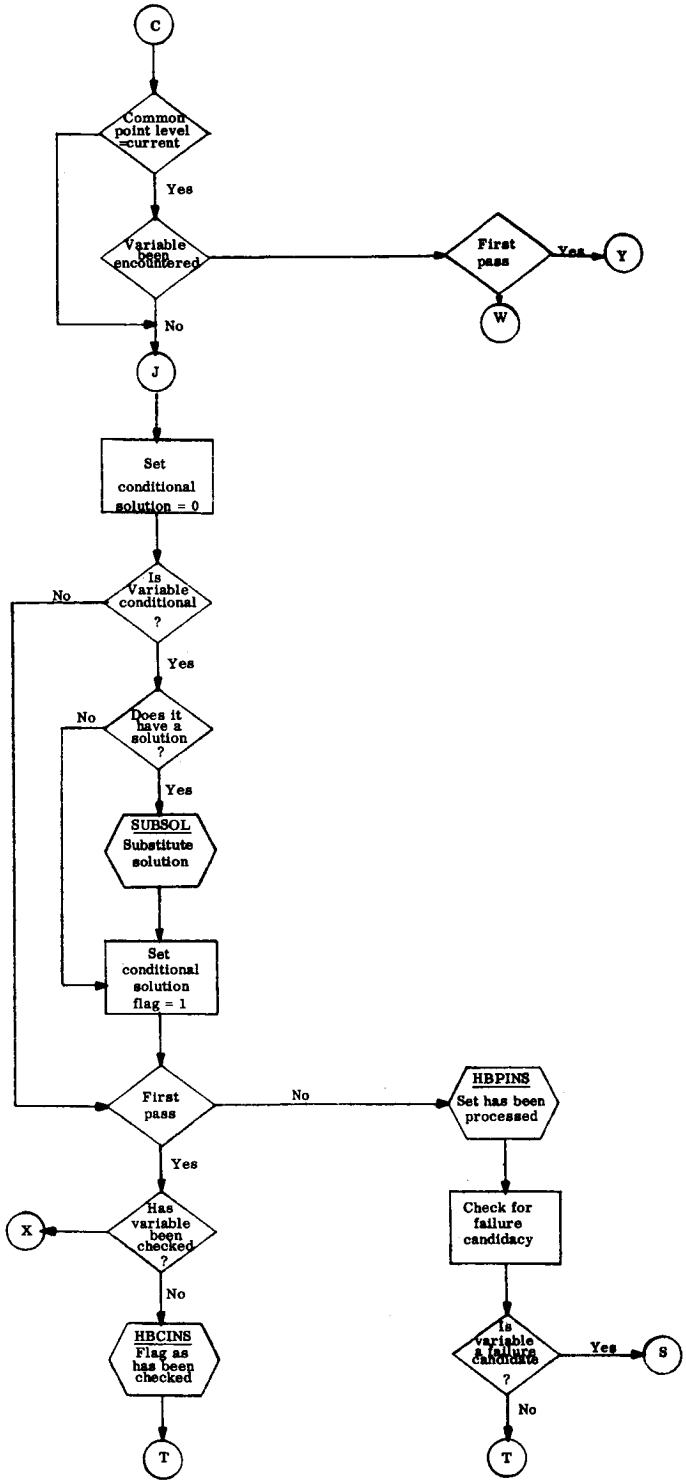
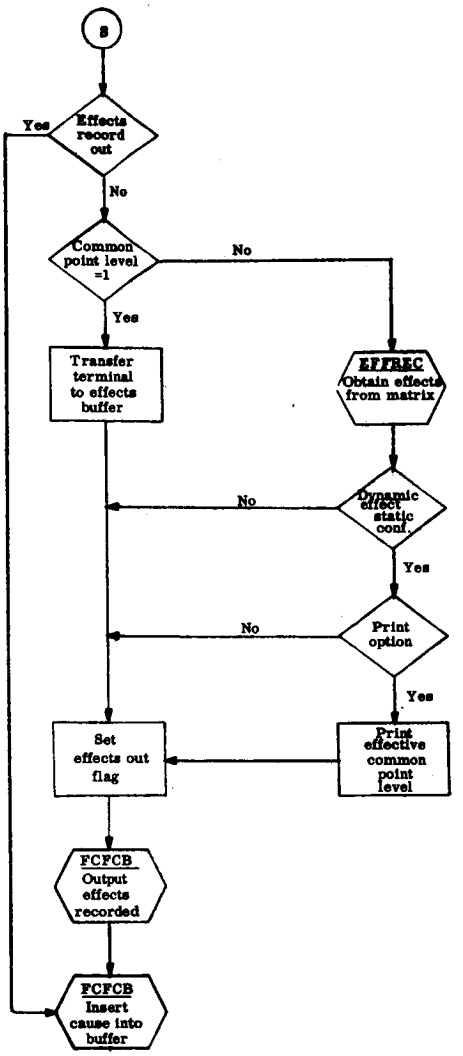


Figure A-9. DOCASE (3)



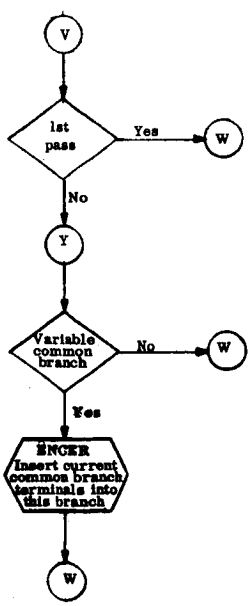


Figure A-10. DOCASE (4)

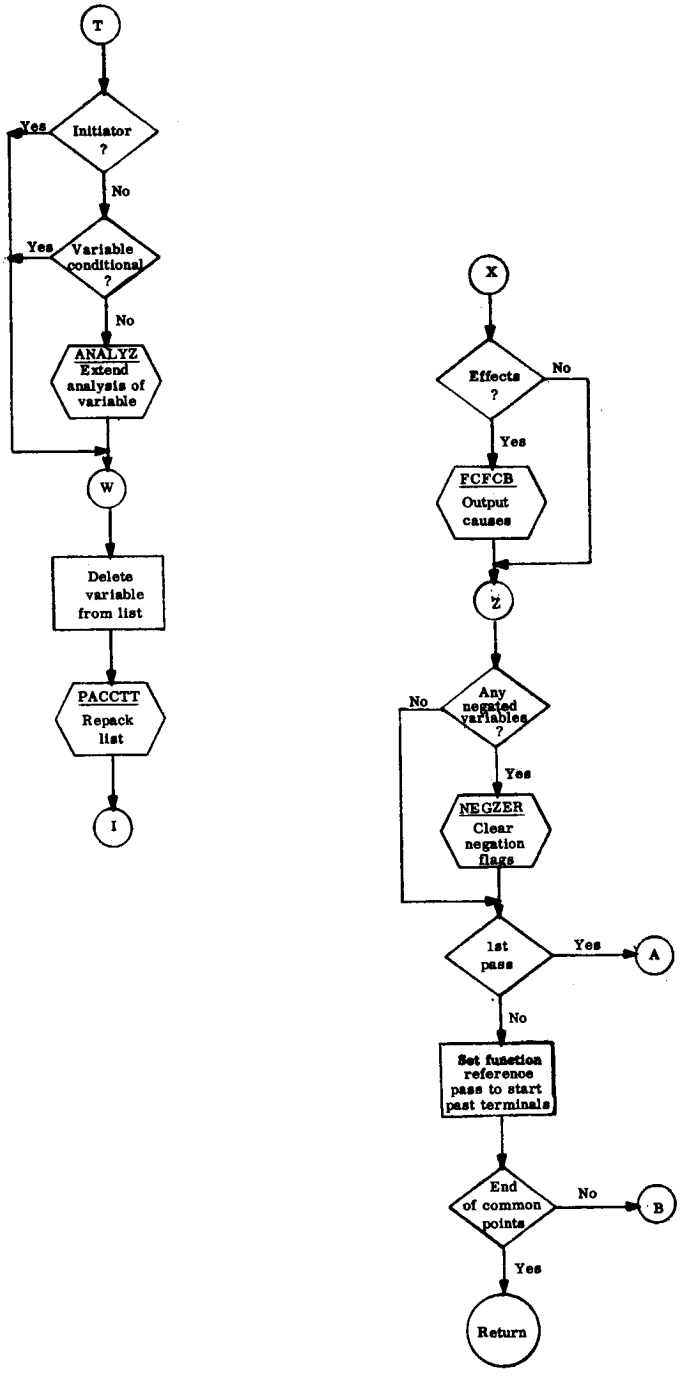


Figure A-11. DOCASE (5)

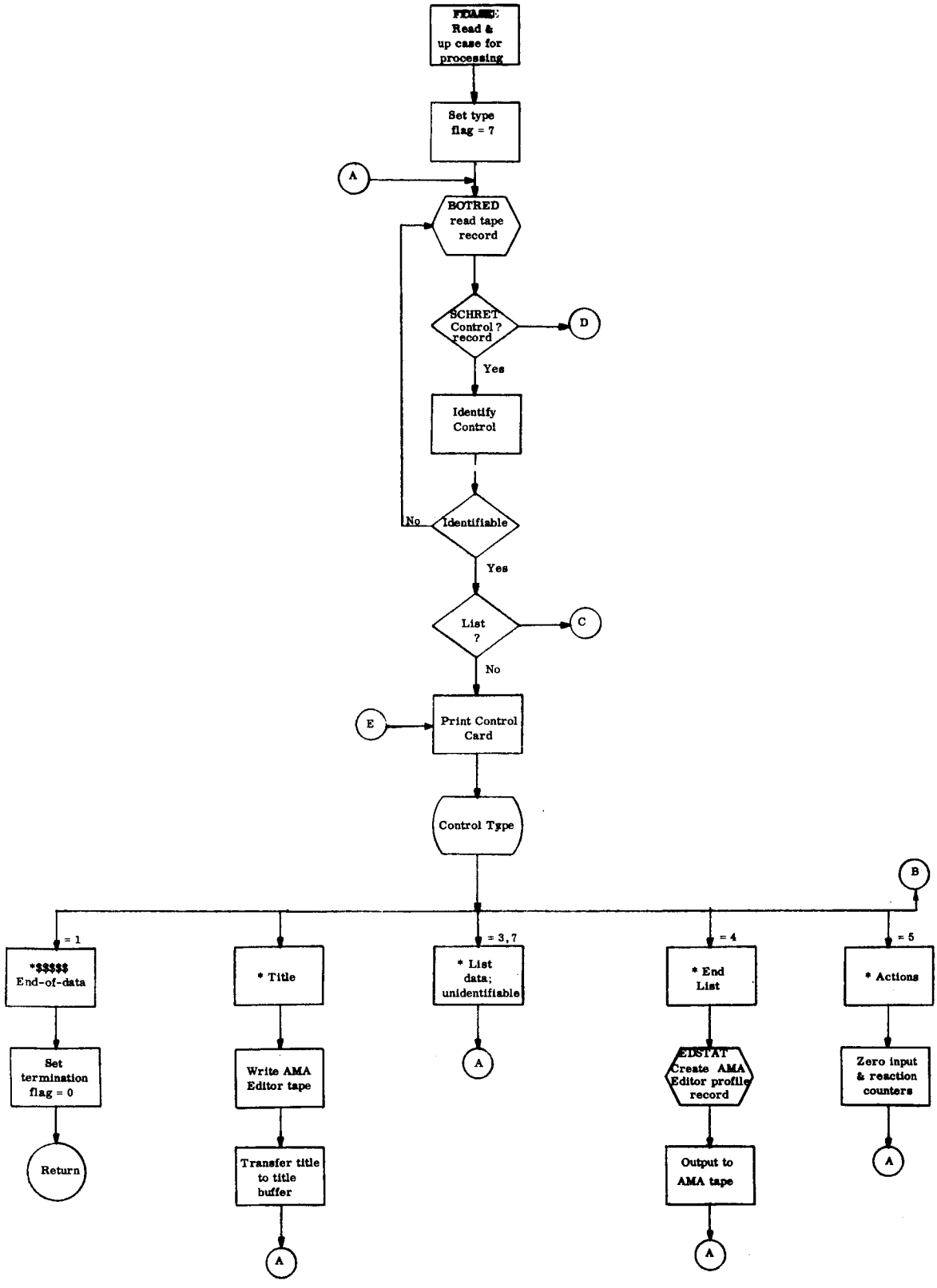


Figure A-12. FCASE (1)

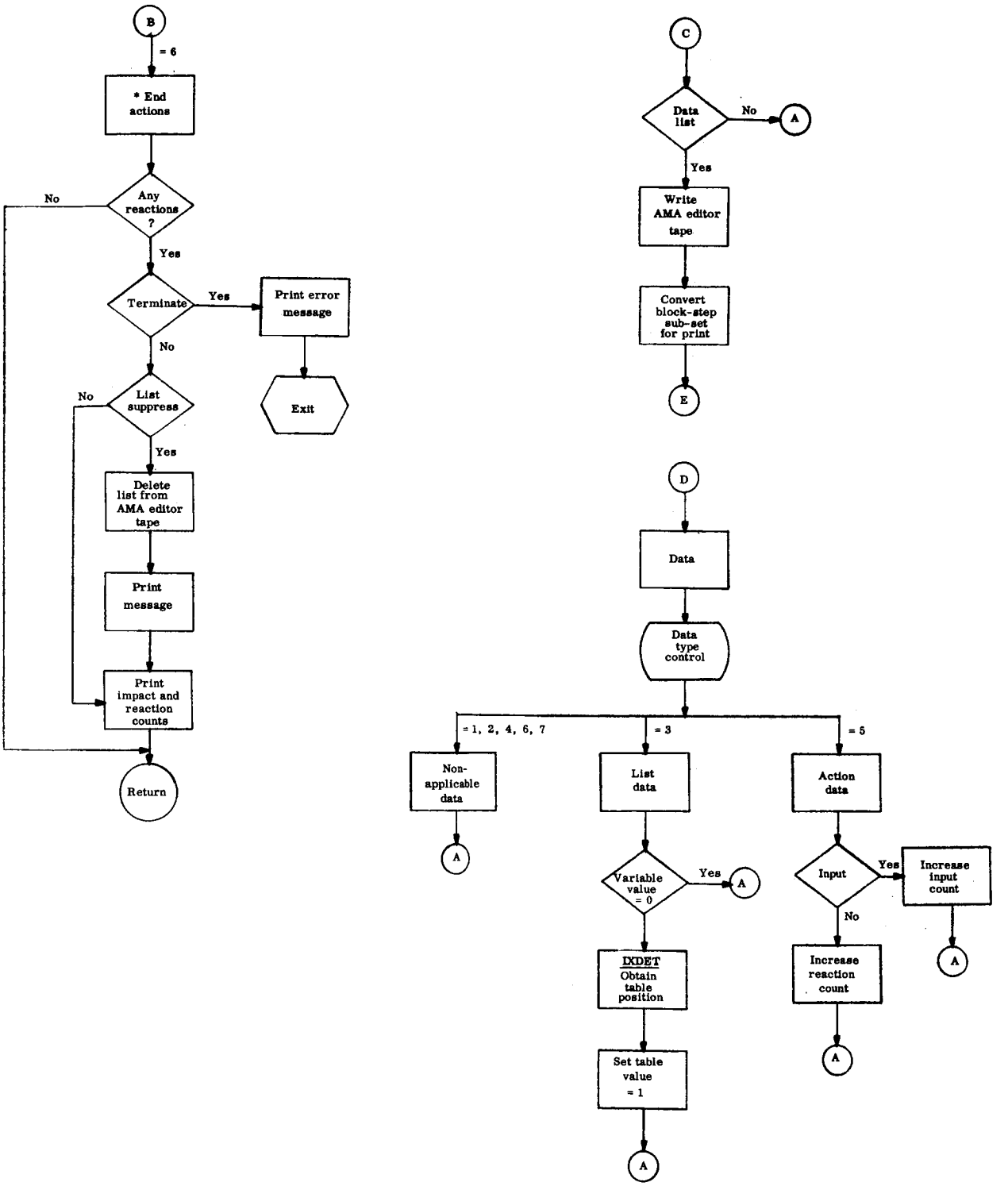


Figure A-13. FCASE (2)

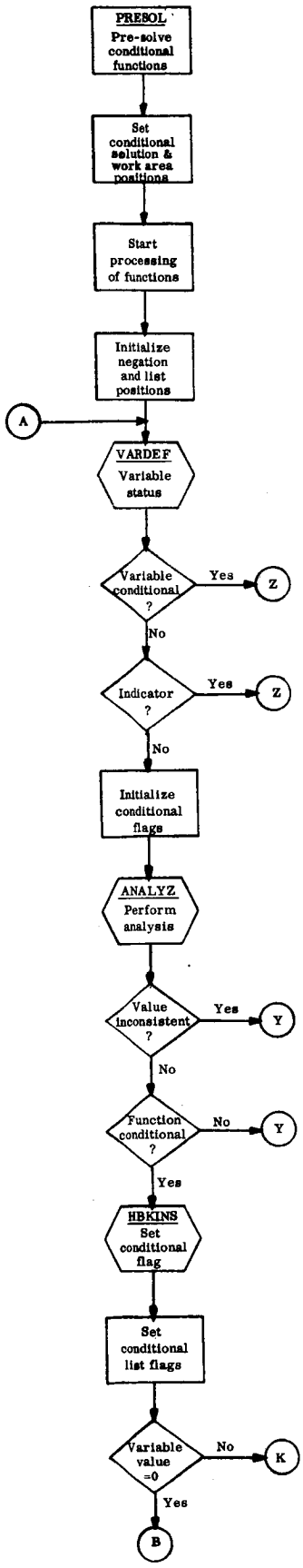


Figure A-14. PRESOL (1)

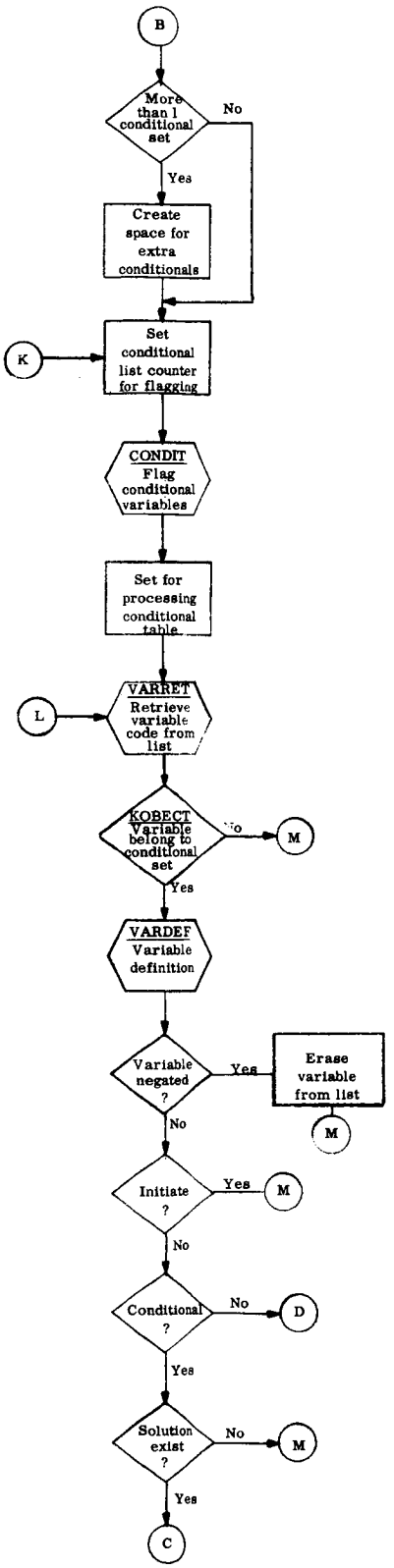


Figure A-15. PRESOL (2)

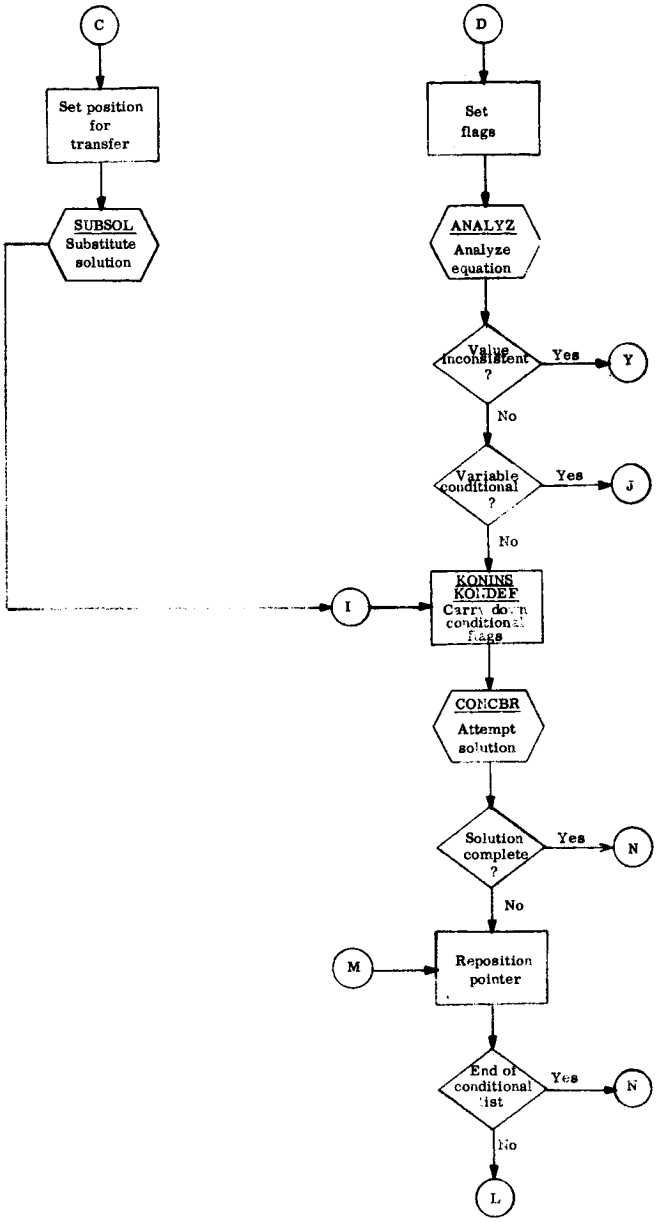


Figure A-16. PRESOL (3)

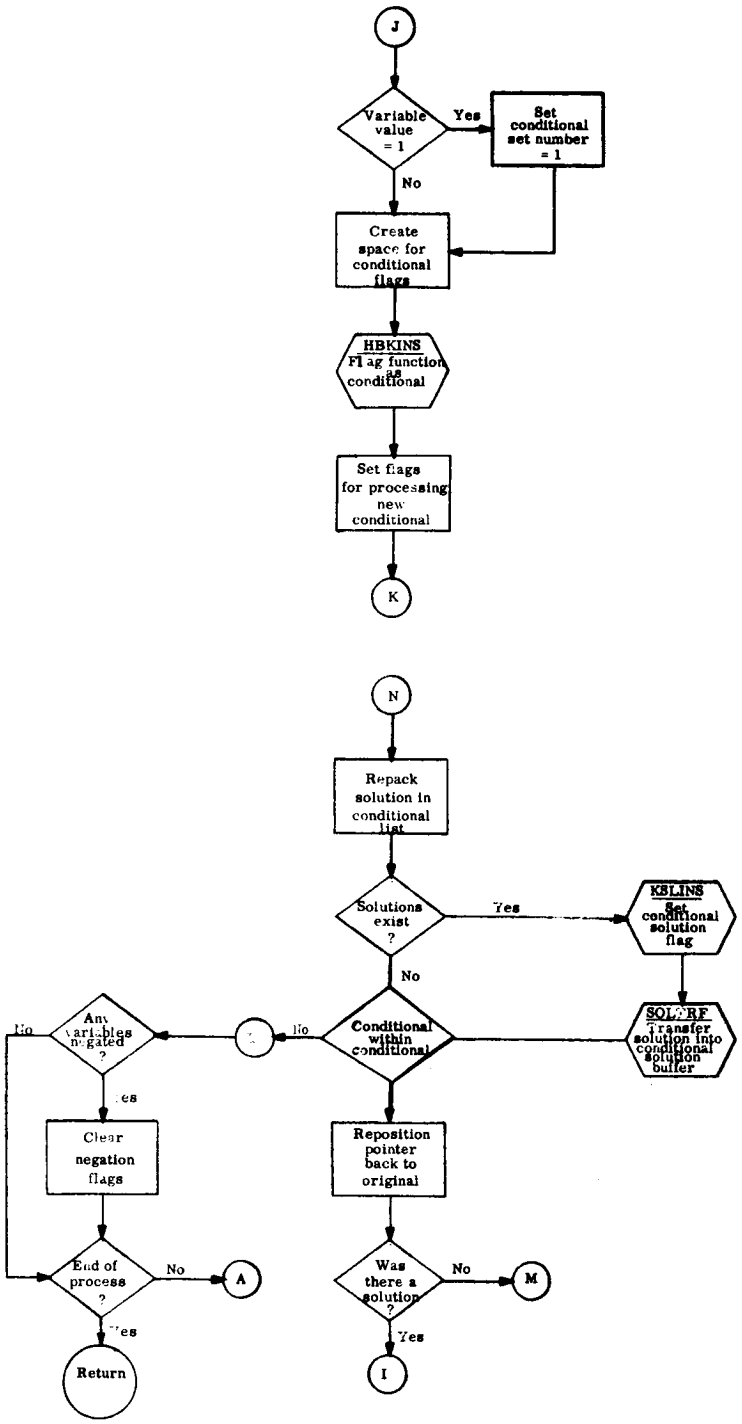


Figure A-17. PRESOL(4)

APPENDIX B: DATA MAPS

I. DATA WORD PACKED FORMATS

A. Equation Packing

Variable names are reduced to 15 bit binary codes, representing their index codes, and operator to 6 bit binary codes by the DT&C program (3843A). Each equation's packed format begins a new word with the packing left justified in all words. The codes which represent the operator codes, used by the AMA program are:

70 - * , and operator

71 - + , or operator

72 - / , not operator

76 - . , equation termination

All equations are packed in one table in ISTORE. No identification separates the equations. Reference is made to the table by way of the first address, through the function reference table, which gives the left-hand member of the equation. Termination of an equation's processing occurs when the processing program encounters the 76 code.

B. Function Reference Data Item

All thirty-six bits of the 7090 word are used for each entry in the table. Significance of the bits are:

S - value of variable (- for 1, + for 0)

1 - if 1, the function is conditional and has a solution
if 0, the function is conditional, it has no solution

2 - if 1, the variable has been processed on previous encounterance at a particular branch level

3, 4 - Failure candidacy

0 - not a failure candidate

1 - failure candidate in one's state

2 - failure cnadidate in zero's state

3 - failure candidate in either state

5 - if 1, variable has been encountered through lower common point level

6-17 - variable's common point level

18 - if 1, the variable has been encountered as a negation of an equation's analysis

19 - if 1, same function of bit 2, but used on prepass

20 - if 1, function is conditional

21-35 - position within the equation table where this variable's equation occurs. If 0, the variable is an initiator.

C. Encountered Common Branch Point Matrix

1. Position

if N is the number of terminals, the number of words assigned in this table to a common branch is I = (M - N) * I + 1.

2. Bit Significance

Since the DI's are set up to be the first variables in system definition, they then will have the first index code assignment. Once the position for a common branch has been determined the terminals which it affects can be determined by the implied position of its corresponding bit. Thus, the sign position indicates the DI whose variable index code is 1, bit 35 represents the DI whose variable index code is 36, bit 4 of word 2 indicates the DI whose variable index code is 41, etc.

D. Conditional Processing Variable Format

1. Function

S - position is set to 1

1-10 - number of branches in function

11-20 - identification of conditional

21-35 - conditional variable, position within function reference

2. Branch member

S -

1-10 - branch to which variable belongs

11-20 - identification of conditional to which variable belongs

21-35 - variable, position within function reference

3. Solution

Before entrance into solution storage, a solution variable consists only of bits 21-35 containing the variable position code. All others are zero.

II. COMMONS

A. Major common is /STREF/ which contains the table storage

1. /STREF/IFREE, IEND, ISTORE (18500)

IFREE - position of first word beyond permanent storage

IEND - contains the value of the maximum limit of ISTORE, currently 18500

ISTORE - table and work area

2. Configuration of ISTORE

a. Permanent storage

1) Equation table

ISTORE (IEQT) to ISTORE (IEQ X)(IEQT = 1)

2) Function reference table

ISTORE (IFNRT) to ISTORE (IFNRX) (IFNRT = IEQX + 1)

3) Encountered common branch matrix

ISTORE (ITVDM) to ISTORE (ITVDMX) (ITVDM = IFNRX + 1)

b. Case storage

1) Conditional solutions

ISTORE (ICONS) to ISTORE (ICTSX) (ICONS = ITVDMX + 1)

2) Working area

a. ISTORE (NAMLIS) to ISTORE (IEND)

1) NAMLIS for pre-solution of conditionals is (IEND - IFREE) * 2/3

2) for unconditional, ICTSX + 1

b. Negation variables

ISTORE (INEG) to ISTORE (IEND)

INEG is incremented backwards from IEND

c. Negation test variable expansion, conditional solution rearrangement

ISTORE (NAMLIX + 1) to ISTORE (IEND)

d. Variable failure, conditional branches list

ISTORE (NAMLIS) to ISTORE (NAMLIX)

e. Conditional solution before transfer, conditional list processing

ISTORE (NNCON) to ISTORE (NAMLIX)

B. Other Commons

1. Unconditional branch processing

/BRHPRS/JFTERM - index code of current common branch

JFUNC - variable position code of current variable

NAMLIS - start of variable failure or conditional branches list

NAMLIX - end of variable failures list, conditional branches list, conditional solutions list

INEG - first address of list of negated variables

2. Conditional branch processing

/CONPRS/KONID - conditional identification

KONDF - conditional solution flag, conditional item counter

KONR - conditional identifier, conditional function code

LEVEL - conditional branch identifier in initial equation processing

NNCON - conditional list processing position value

NBRAN - branch(es) identifier for coded conditional elements

3. Equation unpacking

/EQSKEL/EQUVAL - equation value

NSKEL - number of items in equation unpacked form

EQSKEL(400) - unpacked equation

a) index is set positive

b) operator is set negative

- 1 , *
- 2 , +
- 3 , /
- 7 , .

4. Equation table control

/EQUTAB/IEQT - first position in ISTORE

IEQX - last position in ISTORE

5. Function reference table control

/REFTAB/NTERM - number of DI's in model

IFNRT - first position in ISTORE

IFNRX - last position in ISTORE

IFTER - last position of terminal references in ISTORE

IFCBT - last position of common branch point reference in ISTORE

IFNAV - last position of active variable references in ISTORE

6. Variable definition

/VARSTS/VARVAL - value

VARHBP - has been processed flag

VARHBE - has been encountered flag

VARCAD - failure candidacy code

VARCPL - common point level

VAREQF - equation reference

VARNEG - negation flag

VARHBC - prepass has been processed flag

VARHBK - conditional function flag

7. Conditional solving

/CONSOL/ID - conditional identification on which the solution will be attempted

IN - beginning cell of expansion for solution

NAL - last cell of expansion for solution

NBKON - number of branches in conditional

8. Common level definition

/CPLPRS/ICPMAX - maximum common point level

ICCPL - current common point level of processing

9. Tape definition

/IODEFS/KAMATS - input from DT&C

KTVDM - input from preprocessor editor

KAMAMC - output to AMA Editor

10. Title

/MTITLE/MTITLE(12) - Title from simulator, data set initially to blanks

11. Encountered branch matrix

/TVMTAB/ITVDM - first position in ISTORE

ITVDMX - last position in ISTORE

ITVRW - number of rows in matrix (number of common branches)

ITVCL - number of words to express the terminal effects for each common branch

12. Effect - causes

/EFFCAU/ICFCC - number of words in buffer

KCFC(250) - (a) index codes for either causes or effects

(b) three-letter codes for either causes or effects

13. Simulator input, specification card

/RW/IBIOT(20) - simulator input buffer

LSPEC (4) - input card specifications

14. Conditional solutions

/CONCXX/ICONS - first storage position in ISTORE

ICTSX - last position in ISTORE

APPENDIX C: AMA OUTPUT TAPE FORMAT

The AMA produces one output tape which serves as input to the AMA Editor program.

Format is as follows.

A. 1. First word of each record is the number of words in the record

2. Second word is the BCD identification of the record

B. First record on tape is the title

Word 1: 14

Word 2: *TITLE

Words 3-15: Title

C. Each case of the run has following format

1. Word 1: 14

Word 2: *LIST

Word 9: Block-step substep

2. Word 1: N

Word 2: *STATE

Words 3-N + 1: Bit pattern showing the on/off states of DI's

3. Effect-causes records in sets of two records a piece

Effect:

Word 1: N

Word 2: EFFECT

Words 3-N + 1: index codes of terminal effects

Causes:

Word 1: N

Word 2: CAUSES

Words 3-N + 1: index codes of causes

These records are repeated until next *LIST, or end of tape.

D. Tape may be multiple reel. Each end-of-file causes the call for a new tape. The final tape has the format:

Word 1: 14

Word 2: EOFEOF

PART TWO

AUTOMATIC MALFUNCTION ANALYSIS EDITOR PROGRAM

AUTOMATIC MALFUNCTION ANALYSIS EDITOR PROGRAM

AUTHOR

D. R. Diaddigo
General Dynamics/Convair
Scientific Programming and Analysis
31 March 1966

PURPOSE

The results of automatic malfunction analysis are stored on magnetic tape in separate and intermixed records of failure effects (restricted to DI's) and possible failure causes. Effects and causes are specified using the Discrete Network Simulator internal index codes. The function of the edit and tape generator is to:

1) Correlate internal codes with DI number to produce search keys,

2) Perform the translation between the internal codes and the original model names to produce the lists of failure causes,

3) Reorder the data based on search keys for direct search by a RCA 110 program and create the codes for correlating the effects with the cause lists, and

4) Produce tapes for RCA 110 containing the search keys, correlation codes, failure causes lists, and a hard copy print of the failure causes list.

RESTRICTIONS

1) Program must run on 7090 with IBJOB systems capability;

2) In addition to system input and output units, five magnetic tape units for special input/output are required.

STORAGE

Program/Subprogram Name	Function
1) AMAØUT	Driver
2) CAUPAX	
CAUPAK	Internal packing for failure causes
CAUUPK	Internal unpacking
3) CONVRT	Binary to BCD conversion
4) DIAUX	
DITRAN	Identification and translation of DI number in model
DIPART, DISEP	Auxiliary routines for search key creation and manipulation
5) DINT	Driver to translate DI's, set up core for proper correlation with AMA data
6) DIØN	Create on/off state records for DI's in model for each BLOCK, STEP, SUBSTEP
7) DNSINP	Control input/output of AMA generated data
8) DTCINP	Control input/output of translation between internal codes and model names
9) EFFCØM	Check and modify for redundant failure effect data
10) EFFDET	Process failure effect records, creating internal coded forms and partial search keys
11) EFFMUL	Control storage of multiple failure effects

STORAGE (Continued)

12) ERRMES	Error message prints
13) MALPRT	Creation of failure causes printed lists and tape
14) MALSTR	Core storage and control of failure causes data
15) MALTAP	Buffer storage for failure causes lists
16) MALTEM	Intermediate tape I/Ø, preliminary to final translation of failure cause lists
17) MALTRS	Auxiliary routine to control internal coding of failure causes
18) MDIAUX	
MDITRS MDIMSK MDIREP MDIADX MDIUPO MDIMST MDISEP	Auxiliary routines to control internal coding and storage for multiple effects
19) MIDØUT	Create final search keys for multiple failure effects
20) NAMMAL	Create case storage for original model name translation
21) NAREPK	Auxiliary routine to produce proper packing for RCA 110 failure causes model names
22) NØNPX	Control of creation of static DI information records
23) PRØMAX	Auxiliary routine for DIØN for bit manipulation
24) RCA110	Buffer storage and tape control for RCA110-AMA tape

STORAGE (Continued)

25)	RCAPAX	Data packer for RCA110 word format
26)	RCASWT	Control for multiple RCA110-AMA tapes
27)	RCATAX RCAØPN, RCAWRT, RCAEØF MALWRT MALEØF STARED STACPY	Input-output IOCS routines for RCA110 tapes and static data
28)	SDIAUX SDCNT SDIMAL SDIMST SDUNP	Auxiliary routines for internal core storage and coding of single failure effect data
29)	SDIØUT	Control and arrangement of single failure effect data for output
30)	UNITS	File assignments for Fortran logical units

USE

The program operates in three modes, with corresponding changes in tape requirements. The selection of the operating mode is controlled by the first data card, the configuration of which is:

Mode 1:

Col. 1-6 contain the word ACTIVE
7-12 number of blocks in run
13-72 blank

This mode assumes that no static data is to be included on the malfunction set data tape, so that no merge is attempted.

Tape requirements are:

Fortran Logical	System Function	Tape
11	A(1)	Index-code Name Dictionary
12	A(2)	AMA effect-causes tape
13	A(3)	RCA110-Malfunction Sets
14	B(1)	RCA110-AMA
2	UT2	Intermediate Tape for temporary storage of internal malfunction set codes

Mode 2:

Col. 1-6 contain the word STATIC
7-12 number of blocks in run

This mode is used when a data tape is to be created for later merge with the RCA110 tapes.

Tape requirements are:

Fortran Logical	System Function	Tape
11	A(1)	Same
12	A(2)	Same
14	B(1)	Static AMA data and associated malfunction sets
15	B(2)	Intermediate
2	UT2	Same

Mode 3:

Col. 1-6 contain the word MERGE
7-12 number of blocks in run

This mode is used to produce RCA110 tapes with active DI AMA data and static data combined with malfunction sets.

Tape requirements are:

Fortran Logical	System Function	Tape
11	A(1)	Same
12	A(2)	Same
13	A(3)	Combined malfunction sets and static data
14	B(1)	RCA110-AMA data for active DI's
15	B(2)	Previously created (mode 2) static data
2	UT2	Same

Following this card an identification card is added, which is transmitted to the RCA110 tapes.

Cols. 1-66 Identification

METHOD

The procedure followed by the program is:

1) Set up control parameter for ACTIVE, STATIC or MERGE runs.

2) Read identification card.

3) Ready tapes.

4) Develop and place identifying codes on RCA110 tapes.

5) Initialize malfunction set core storage, identify and correlate DI numbers and codes for test procedure. If merge runs, place static data on malfunction set tpae.

6) For BLØCK loop, maintain count of BLØCK and compare with number of blocks in run to establish exit point.

7) For each SUBSTEP - the procedure used

    a) Create BLØCK, STEP, SUBSTEP identification

b) Read effect record and determine DI configuration and storage positions.

c) Read causes record and store malfunction set or determine if it has already been processed (Malfunction sets pertain to entire test procedure) and store the internal set number with the appropriate effect storage. Translation between the internal set number and the external number is maintained.

d) Repeat b) & c) until next SUBSTEP or end-of-tape is encountered.

e) At end-of-substep

(1) Output single DI failure effect, with malfunction set number translated

(2) Perform reordering cycle for multiple failure effects, output search keys and translated malfunction set numbers

(f) Repeat a) to e) until end of tape is encountered.

8) At the end of AMA data input

a) End RCA110-AMA data input

b) Output the collected malfunction sets on a temporary tape

c) Read Index-Code-Model Name Translation tape

d) Re-read the temporary tape, translate the index coding of the malfunction set to model names and place the sets on tape and print on system output unit. (If static run, send to AMA data tape).

e) If merge run, copy malfunction sets from static tape onto the RCA110-malfunction set tape.

f) End tape.

9) End run.

APPENDIX A: PROGRAM PROCEDURES AND FLOW CHARTS

Reference to data items are made to Appendix B outlining data storage and the labeled commons.

The main program driver outline appears under method, and the flow charts appear at the end of this section.

Subroutines are discussed in alphabetical order with an explanation of their options and techniques and their interrelationship with other programs through labeled common.

ROUTINES

2. CAUPAX - Two entries

   CALL CAUPAK - Pack index codes indicating the causes from /DNSDAT/ into /DNSDAT/ under DNSCNT control.

   CALL CAUUPK - (Address of malfunction set to be unpacked, address of array to unpack into)

3. CØNVRX - Used as function to connect binary to BCD

   X = CONVRT (NUM), number is returned in accumulation right-justified with leading blanks.

4. DIAUX - Auxiliary DI number processing routines. Three entries:

   CALL DITRAN

   Words from /NAME/ searching for DI configuration as DIXXXX, where XXXX is DI number. Number is connected to binary (B17) and stored in DIBIN of /DICØN/. Its word and bit configuration is computed on the basis of its binary number, assuming that word 1 of a 63 word profile contains DI's 23-0 in that order, word 2 contains DI's 47-24, etc. This is placed into DICØN of /DICØN/.

CALL DIPART (Address of 63 word profile)

Routine is entered assuming that DICON of /DICON/ has the complete word configuration (bit and word position) for oring into 63 word profile.

CALL DISEP - Uses /DICØN/ Common

DICØN contains DI profile configurations and the routine separates it into word position (B35), stored in DIWRD and bit configuration, stored in DIBIT

5. DINX - Processes Index-Name tape for DI's. Routine assumes DI's are the first names on the tape and the processing is terminated on first non-DI.

CALL DINT -

Routine passes name through /NAME/ to DITRAN and returns the Binary and profile configurations through /DICØN/ which are then stored in the single DI section of /AMADAT/. The active DI code profile is placed in PARDIS of /PARDIS/.

6. DIØX -

CALL DIØN - Routine is called upon the reading *STATE record from failure effect-causes tape.

Since the DI's are assumed to start with index code 1, the program loops from 1 to the terminating index code retrieving from the on/off state words in /DNSDAT/ the code for each DI through routine PRØMAS. If the state is '1' DIPART is used to store the bit into the proper word of DIØNØF of /PARDIS/.

7. DNSINP

CALL DNSINP - Reads AMA failure effect-causes tape and determines the type of record.
Data is read into /DNSDAT/
Count in DNSCNT and succeeding data into /DNSDAT/.
Identification of control word (DNSDAT(1)) is placed in AMATYP, whose values are

1. *TITLE
2. *LIST
3. EFFECT

4. CAUSES
5. EØFEØF
6. *STATE

One terminating error controls the inputting of unidentifiable data.

8. DTCINX

CALL DTCINP(N) - Reads data from index-name tape.

N = 1 Identify record, setting DTCTYP of /NAME/ to 1, for *NAMES control
2, for *REFERENCE control
3, for no id

N = 2 Transmit name or end names flag to calling routine. Name buffers are read in /DNSDAT/. Names are transferred through /NAME/ with name length (words) in NAMLEN, index code in INDEX, DTCTYP set to 3, and the name in NAME.

9. EFFCØX

CALL EFFCØM

Routine works from /FAILDI/ common. If a single effect is involved it checks the appropriate code words of the single DI block of /AMADAT/ based on the NCØD value. If a multiple effect is involved it checks through the chain based on the NDI value. If a duplicate is found in either case IDUP is set negative; otherwise, it is positive.

10. EFFDEX

CALL EFFDET

Works from /DNSDAT/ reducing the EFFECT record to the proper word configuration for storage into /AMADAT/. No configuration to be compared in EFFCØM and stored (SDIMST/MDIMST) is placed in /FAILDI/ setting NDI to the number of DI's involved, NCØD to the number of words in the multiple DI configuration (or position in section 1 of /AMADAT/ for single DI), and DICØDE with the 63 word profile configuration. (DI's occupying the same word are used together and the vector is ordered by word position number).

11. EFFMUX

CALL EFFMUL - Store multiple DI failure effect configuration. Stores DICØDE into next available words of section 2 of /AMADAT/ and uses MDIUPD to update the chain lists in AMADAT and form the control configuration word.

12. ERRMEX

Error Message routine

Call ERRMES(N), where N is the error encountered. All errors produce an immediate termination from the program and each indicates an error in format/loading of either the index code-variable tape or the AMA tape.

13. MALPRX

Production of malfunction set tape and hard copy print.

Routines operate under control of /MALMAS/ using MALTEM to retrieve the decoded malfunction set number and components, the configuration 2 setup of /AMADAT/ to determine the external names and lengths and NAREPK to reform the name in RCA110 format. Under a static control run it adds 10000 to both tape and hard copy malfunction set number. Under merge control it adds AMA static search keys and malfunction set data onto the Malfunction Set Tape through STACPY. The hard copy print is done directly through this routine.

14. MALSTR - Set up Section 3 of /AMADAT/

CALL MALSTR - malfunction set to be stored is contained in /DNSDAT/ and its length in /MALSET/. It is immediately packed into /DNSDAT/ through CAUPAK routine.

(1) If malfunction set pertains to single DI, it checks directly against the malfunction set setup for the particular DI on last substep. If it has remained the same, no further action; otherwise (2) is executed.
(2) Search the stored malfunction set up to this point (section 3 of /AMADAT/). If this has been stored before /MALSET/ is set to the matching address; if not a new malfunction set is created and the position information of /MALMAS/ is updated.

15. MALTAX

Controls output to malfunction set tape.
If under control of a static run, data is set to RCA110-AMA tape.

CALL MALTAP (N, IDT, ICNT)
N is optional entry

N - 1, write the MALFUNCTIØNS id for tape's malfunction set data.
N = 3, store ICNT words of information from IDT into /IRCA/ output buffer.
N = 2, initialize buffer storage and execute option N = 3.
N = 4, pack according to RCA110 format and write information onto tape.
N = 5, end tape with EØF's

Uses MALWRT MALEØF for tape writing
RCAWRT RCAEØF

16. MALTEX

CALL MALTEM(N)

N = 1, under control of /MALMAS/ the routine unpacks each malfunction set (CAUUPK from /AMADAT/ into /DNSDAT/ and /MACSET/), and places it as a separate logical record on temporary tape (UT2). Each set occupies a separate logical record.
N = 2, read a single malfunction set into core (/DNSDAT/, /MALSET/)

17. MALTRX - Two entries

CALL MALTRS (Address of malfunction set control word)
Picks up control word and separates out the word count and external set value in /MALSET/.

CALL MALCTR

From /MALSET/ and /MALMAS/ form control word for malfunction storage. Control word is in /MALSET/.

18. MDIAUX - Seven entries

CALL MDITRS (Address of multiple DI control word)
From storage in section 2 of /AMADAT/ separate out word count, malfunction set, and control word into /MDIMAL/.

CALL MDIMSK (Configuration for DI, word to place configuration in)
To build up multiple effect but configuration word by oring together bit patterns of same words of 63 word profiles.

CALL MDIREP (multiple DI configuration storage)

To create replacement word configuration for search keys. Operates from /FAILDI/ where configuration of last search key on tape is located and forms the replacement word sequence into /REPLAC/.

(a) Words not directly replaced by the new multiple DI, have zeroing words created for them.
(b) Direct replacements and new additions are stored directly into /REPLAC/.

CALL MDIADX (Address containing a multiple chain reference)

Separates out the multiple chain address into /MDIMAL/

CALL MDIUPD

From /MDIMAL/ create and store the information necessary to maintain chain lists of multiple effects of the same level.

CALL MDIMST (Address of multiple DI control word)

From /MALSET/ store malfunction into a multiple DI configuration control word.

CALL MDISEP (Address of word to decode)
Decode a multiple DI bit-word position control word into /MDIMAL/.

19. MDIØUX - Control of output for multiple DI failures.

(1) Control is on worst case first by searching the third words of section one of /AMADAT/ backwards, and using the chain controls there. First multiple DI under this search is made into the 63 word profile.
(2) Continuing the search on chain and worst case the replacement sequences are formed into /REPLAC/ and transmitted through RCA110 routine.
(3) /FAILDI/ is updated after each replacement sequence to show the current search key which must undergo replacement.

20. NAMMAX

CALL NAMMAL

Create configuration 2 of /AMADAT/. Using failure candidate information of the index-variable translation tape, it determines if name should be stored. If not, the control word is made negative. Uses DTCINP to read tape and /NAME/ to retrieve name and pertinent information.

21. NAREPK

CALL NAREPK (Address of name to pack into RCA110 format, number of words to pack)
Packs into /NAME/ right justifying the words with four characters and deleting trailing words that contain only blanks. The number is modified to the number of words that contain the name after packing.

22. NØNPX

CALL NØNPT - Called during merge run to read and transmit static AMA data and static codes to malfunction set tape, and/PANDIS/ common.

23. PRØMAX

X = PRØMAS (INDEX code - 1).

Retrieve the final on/off state of variable with indicated index code from the STATE record of AMA data tape. Operates on the packed state list (36 variable/word) using implied position. Operates from /DNSDAT/ placing on/off state in AC.

24. RAC11X - General routine to handle writing of RCA110-AMA tape.

CALL RAC110(N,IADD,NCNT)

N = 1, read and write RCA110 control record (from card to tape). If active run also write on malfunction set tape.
N = 2, Initialize buffer and store NCNT items of data from IADD into /IRCA/.
N = 3, Perform storage, without initializing if data to be stored exceeds buffer area of /IRCA/ remaining, fill buffer, pack, write and continue with CØNT record.
N = 4, Pack, and write any data remaining in /IRCA/
N = 5, End RCA110-AMA tape
N = 6, Write RCA110 tape id (previously read under N = 1) on RCA110 tape.

25. RCAPAK

CALL RCAPAK (Buffer, number to pack, number of words after packing)

Each 7090 word upon entrance into this routine is assumed to contain an RCA110A character word right-justified. The routine creates full 6-character 7090 words for direct transmission to tape.

26. RCASWX

CALL RCASWT(1) - Write terminating id on old RCA110-AMA tape and end tape.

CALL RCASWT(2) - Initialize new RCA110-AMA tape rewriting ID and test-block identification.

27. RCATAX - IOCS tape control routine

CALL RCAØPN - open RCA110-AMA tape

CALL RCAWRT - (address of buffer, word count)

CALL RCAEØF - Write end-of-file and set end-of-tape bit

CALL MALWRT - (address of buffer, word count)

CALL MALEØF - Write end-of-file

CALL STARED - (address of buffer, words read)

CALL STACPY - Copy static data onto malfunction set tape.

(1) Terminate copy of MALFUNCTIONS
(2) Terminate copy of end-of-file

28. SDIAUX

CALL SDCNT (ICT) - pass through section 1 of /AMADAT/ setting ICT to the number of single DI failures for this substep.

CALL SDIMAL (Address of single DI control word).
From /AMADAT/ store malfunction set number associated with single DI into /DICØN/.

CALL SDIMST (Address of single DI control word).
From /MALSET/ store address of malfunction set number.

CALL SDUNP (Address of bit-word configuration for DI).
Seperate bit-word configuration into separate word and bit words in /DICØN/.

29. SDIOUX

CALL SDIOUT

Operating on section 1 of /AMADAT/ for single DI effects create search keys for RCAll0-AMA tape.

(1) Translate malfunction set address to external number and create control words.

(2) Determine total count of record.

(3) If static run, add 10000 to external malfunction set numbers.

30. UNITS - File control cards for

(1) INDEX-variable translation tape (from DNS-AMA-DTC).

(2) AMA data (from AMA program).

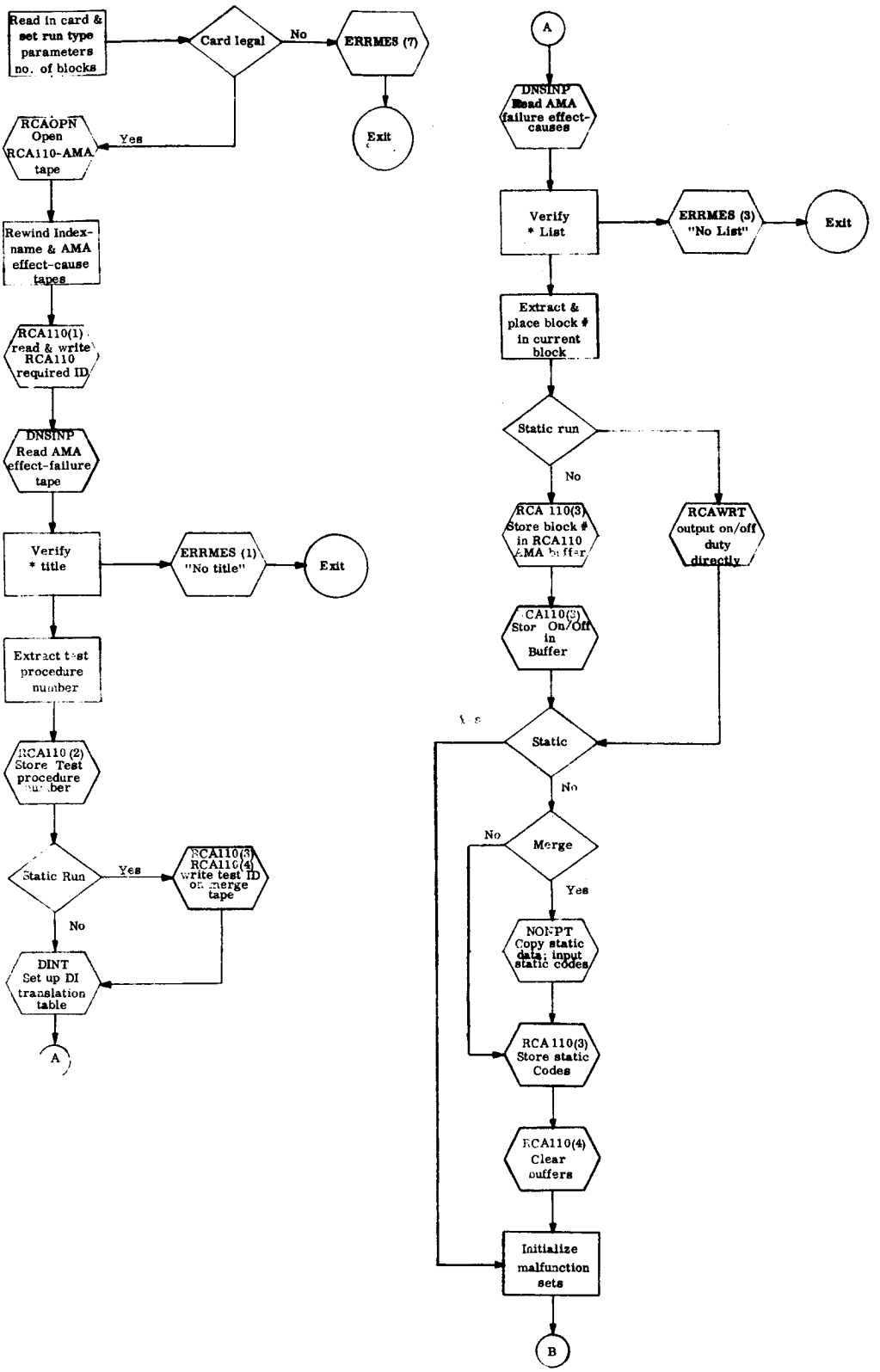


Figure A-1. Main Program (1)

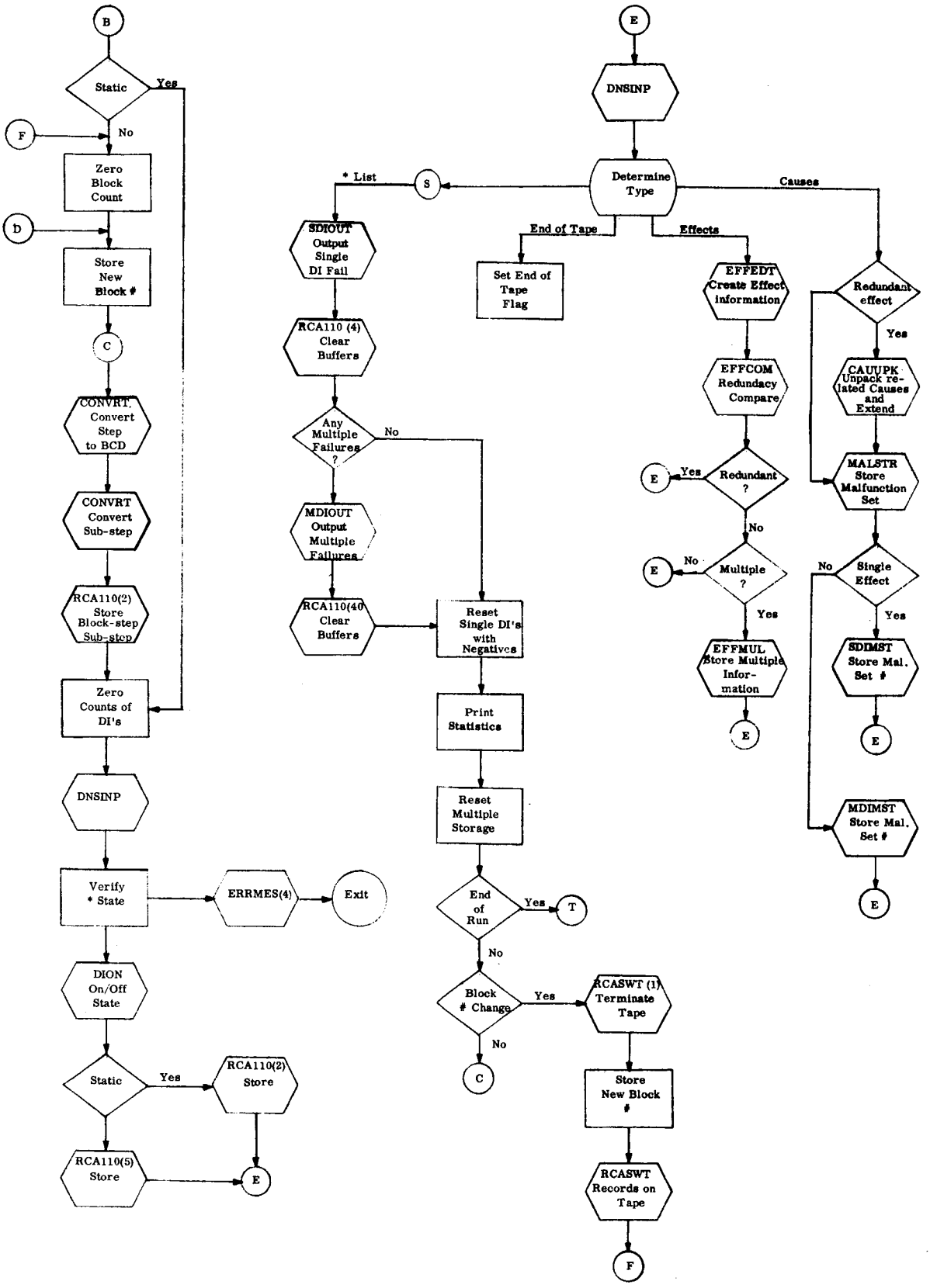


Figure A-2. Main Program (2)

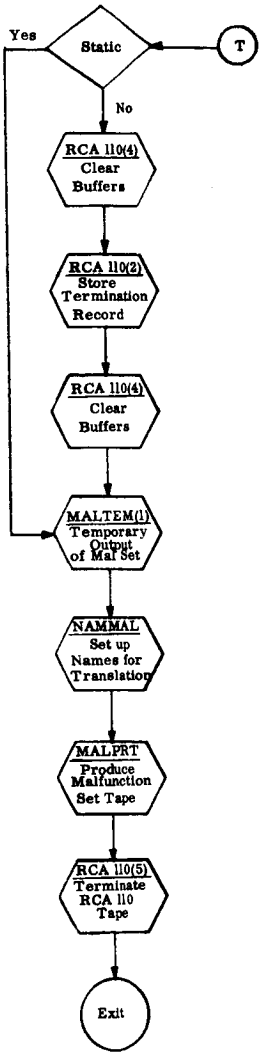


Figure A-3. Main Program (3)

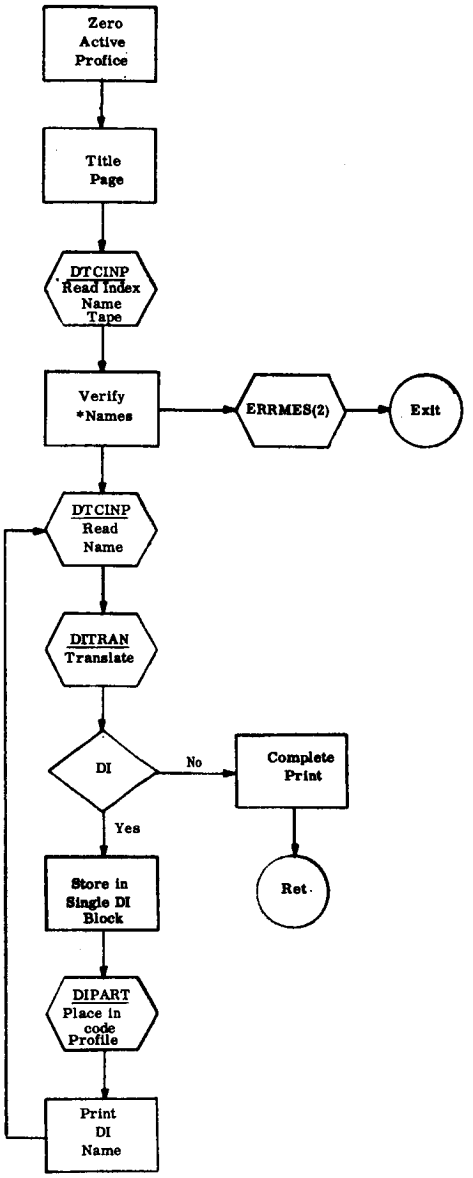


Figure A-4. DINT

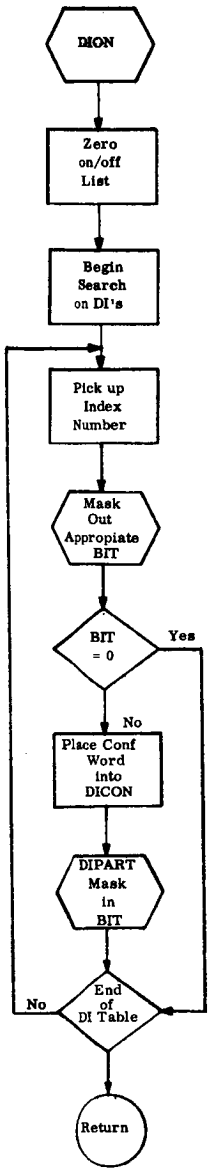


Figure A-5. DION

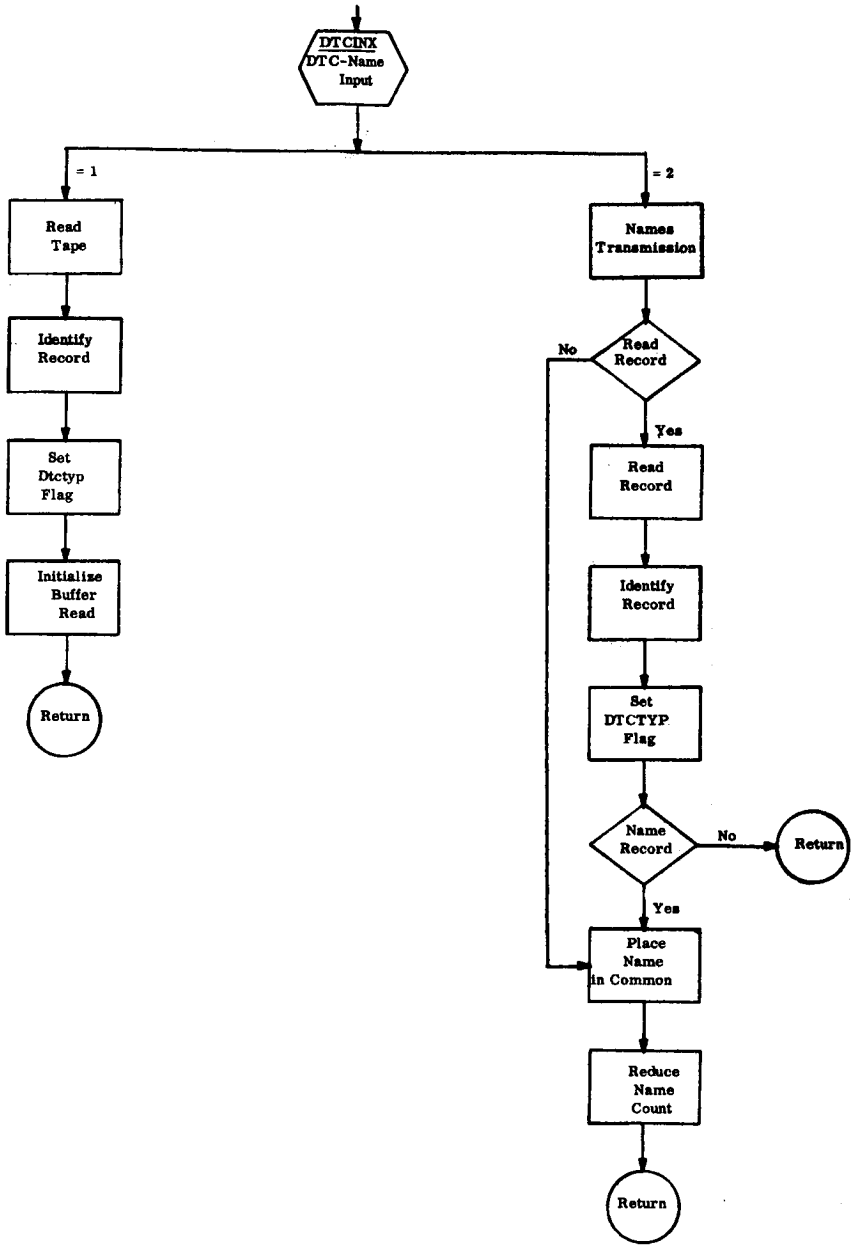


Figure A-6. DTCINX

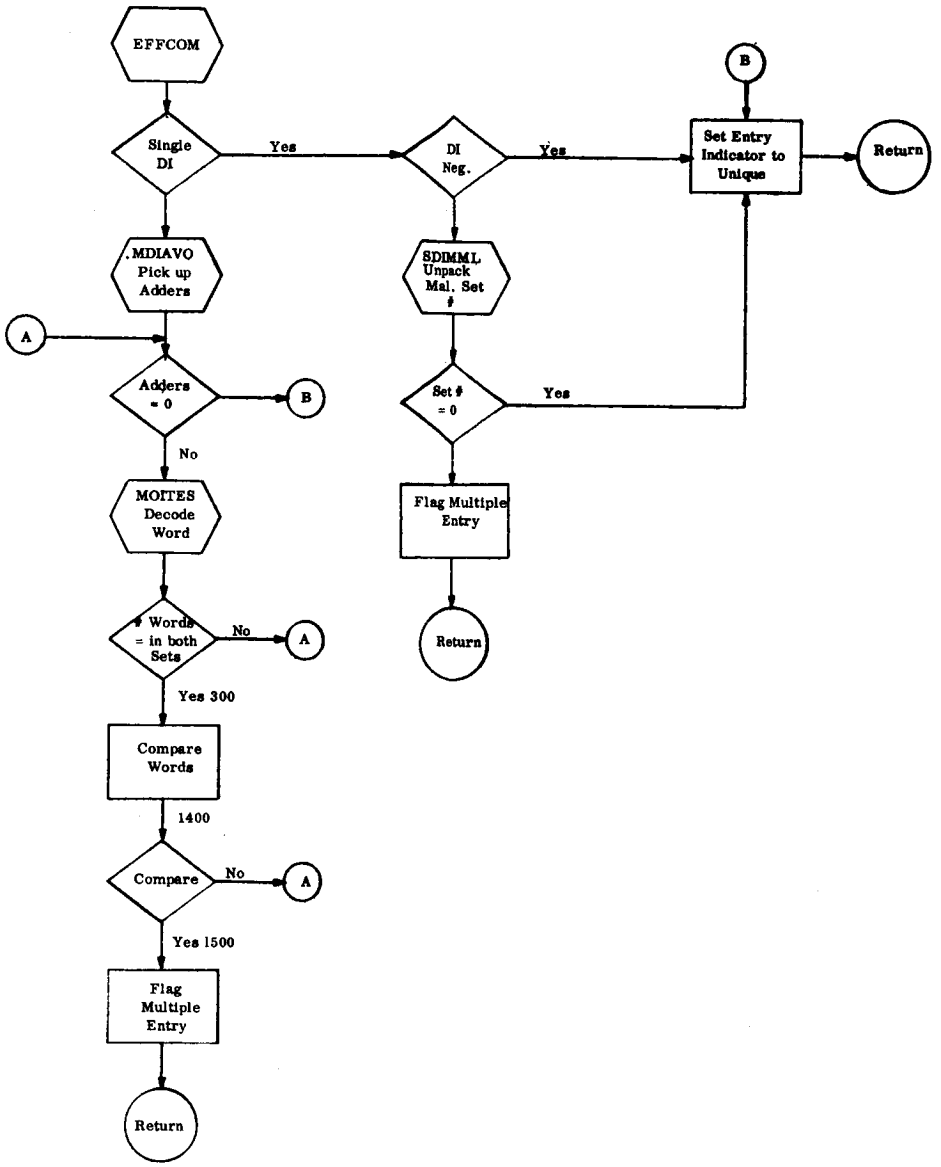


Figure A-7. EFFCOM

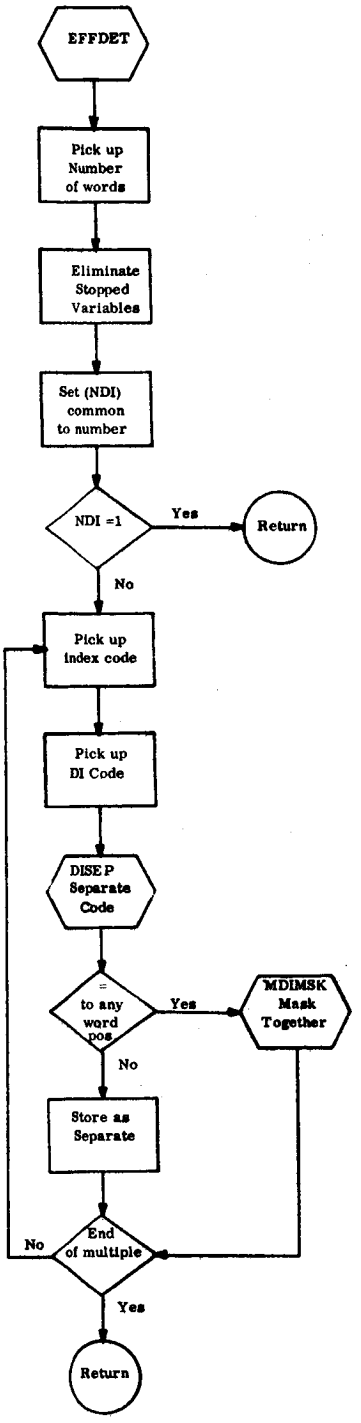


Figure A-8. EFFDET

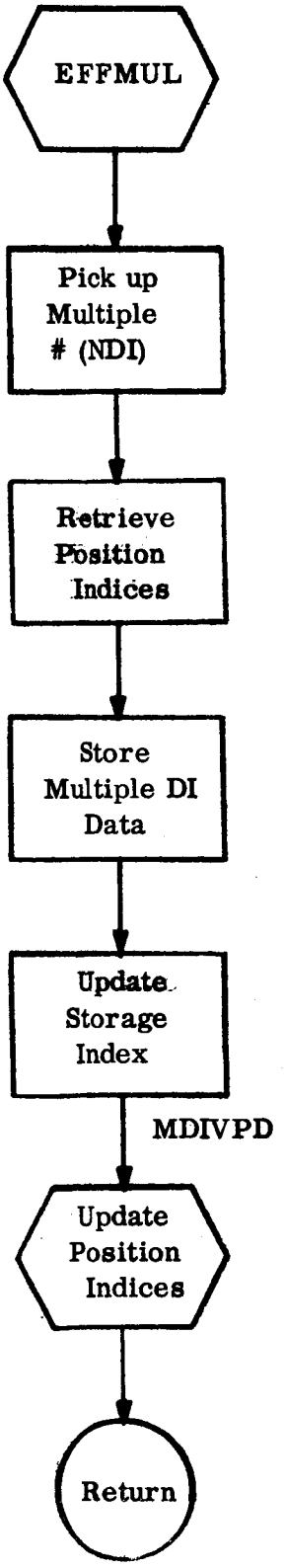


Figure A-9. EFFMUL

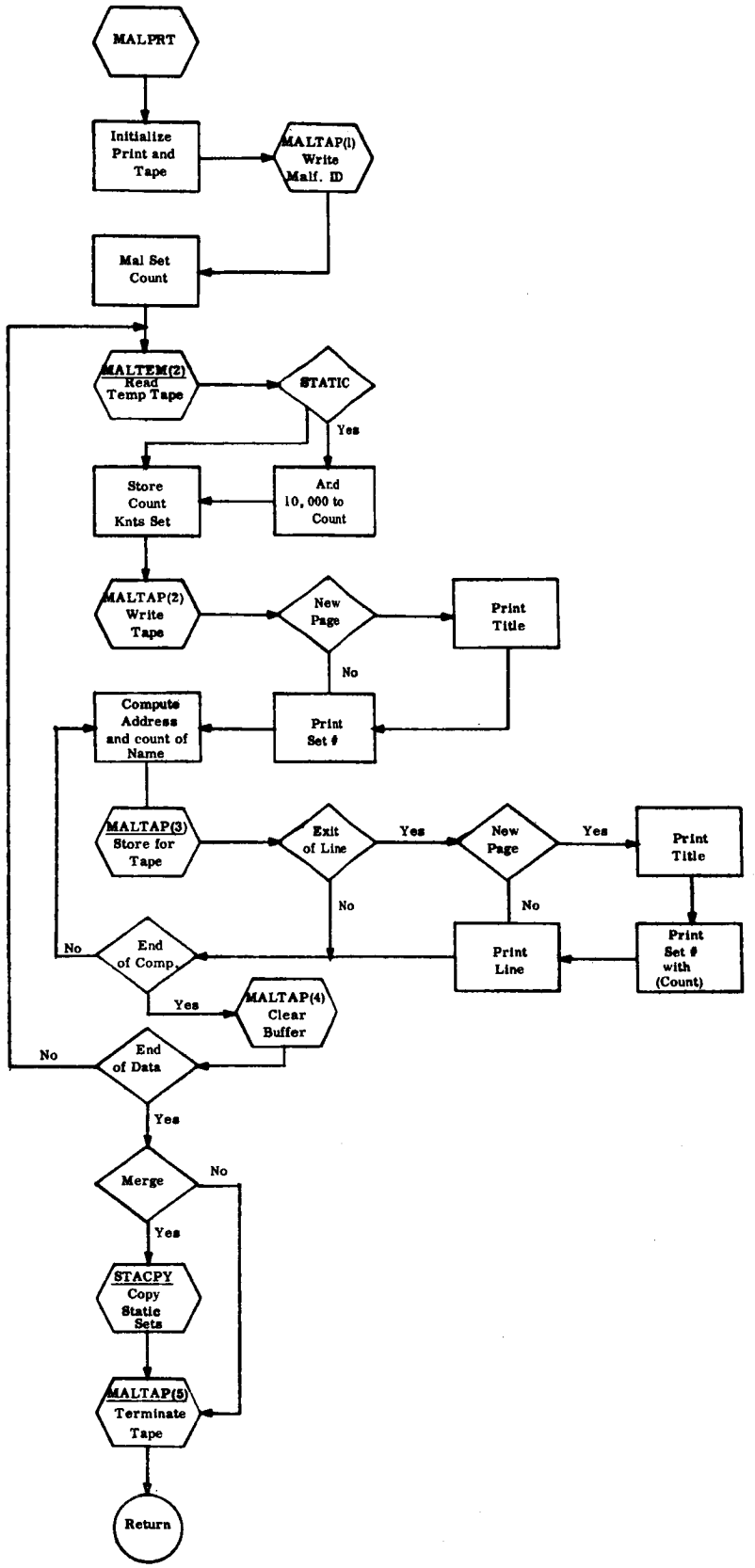


Figure A-10. MALPRT

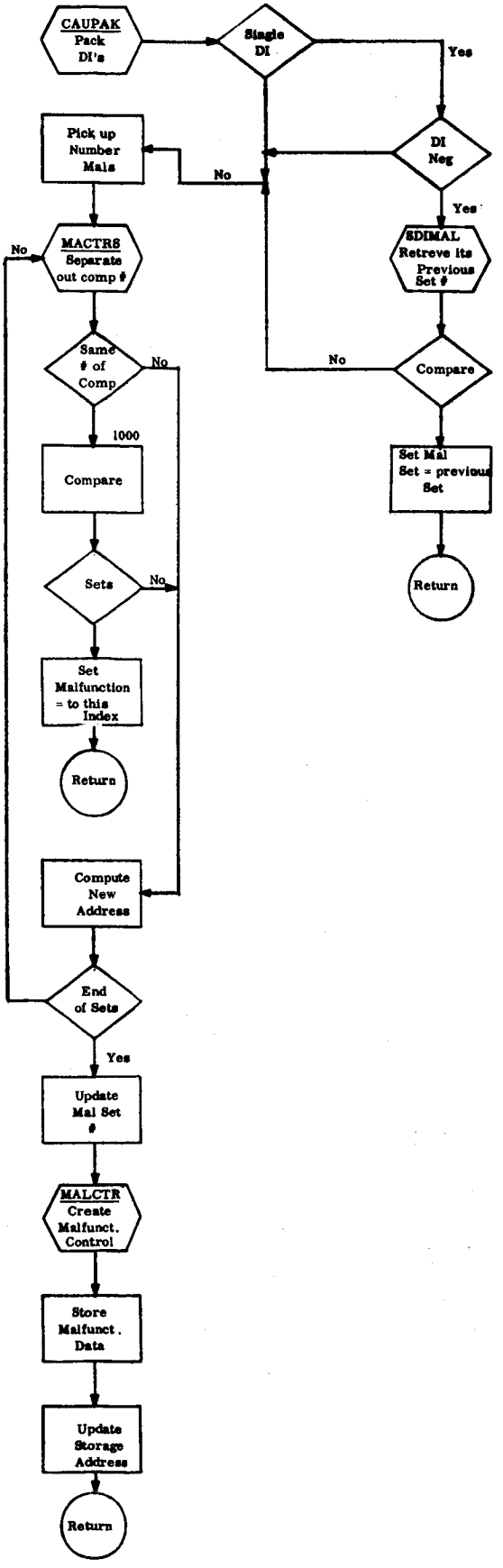


Figure A-11. MALSTR

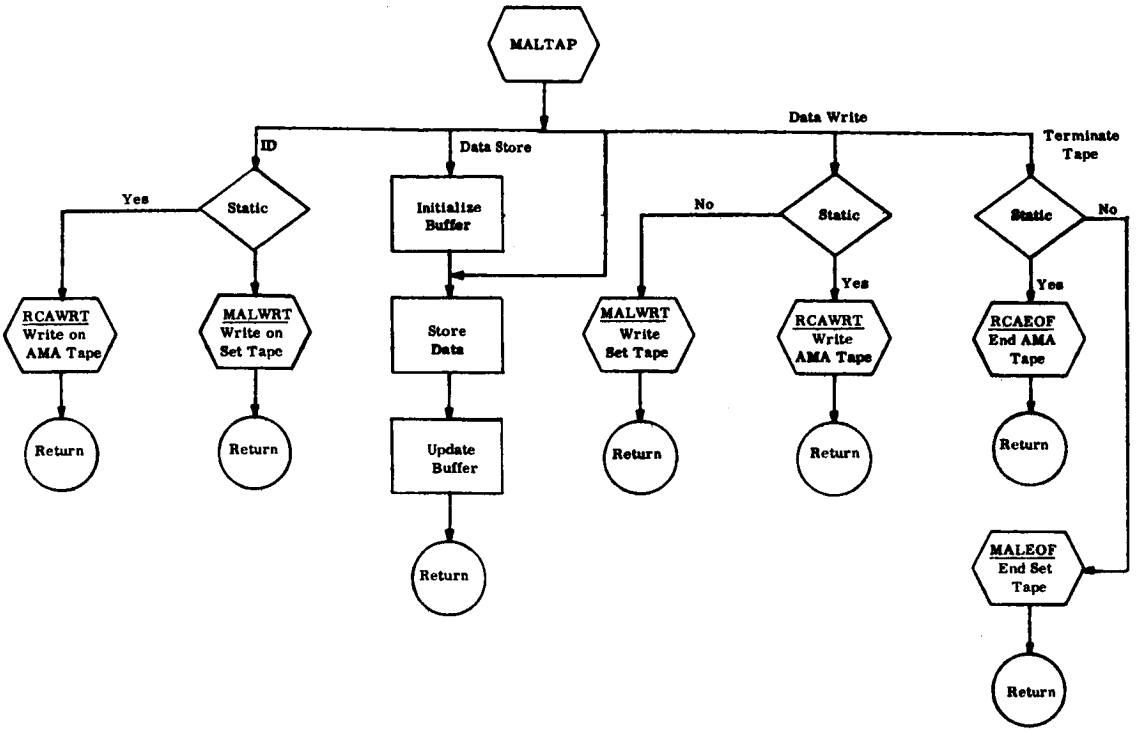


Figure A-12. MALTAP

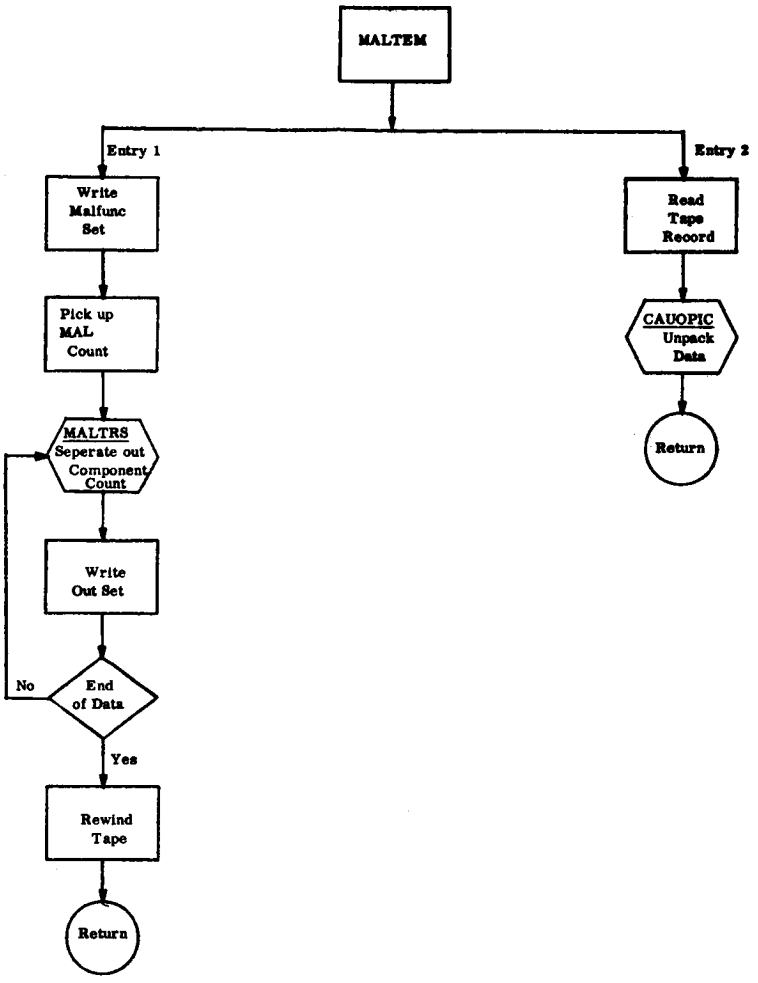


Figure A-13. MALTEM

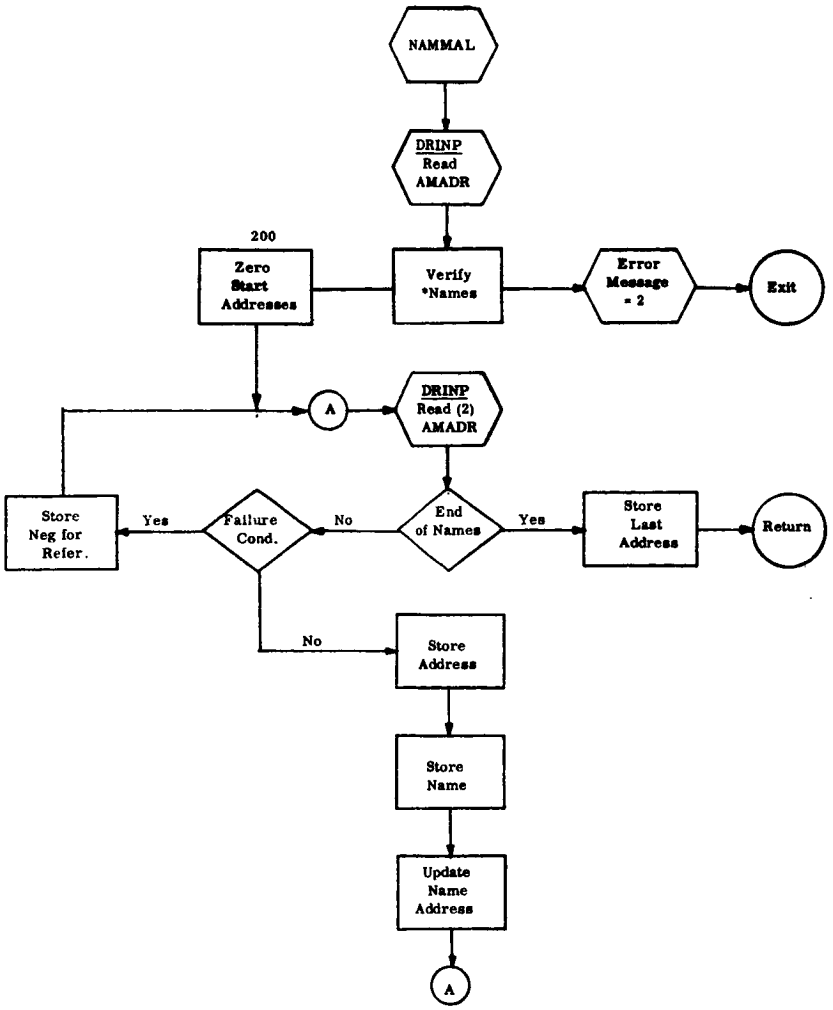


Figure A-14. NAMMAL

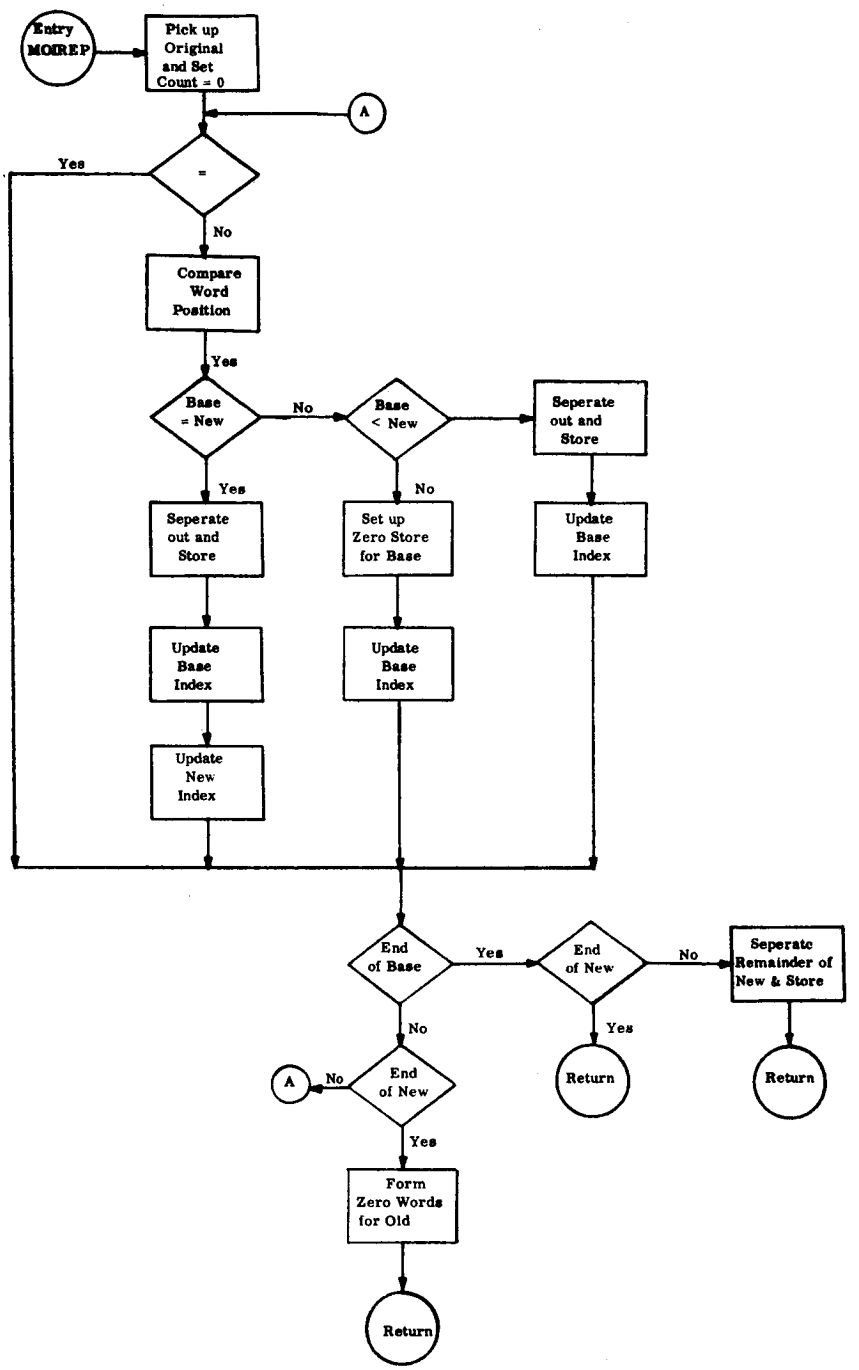


Figure A-15. MOIREP

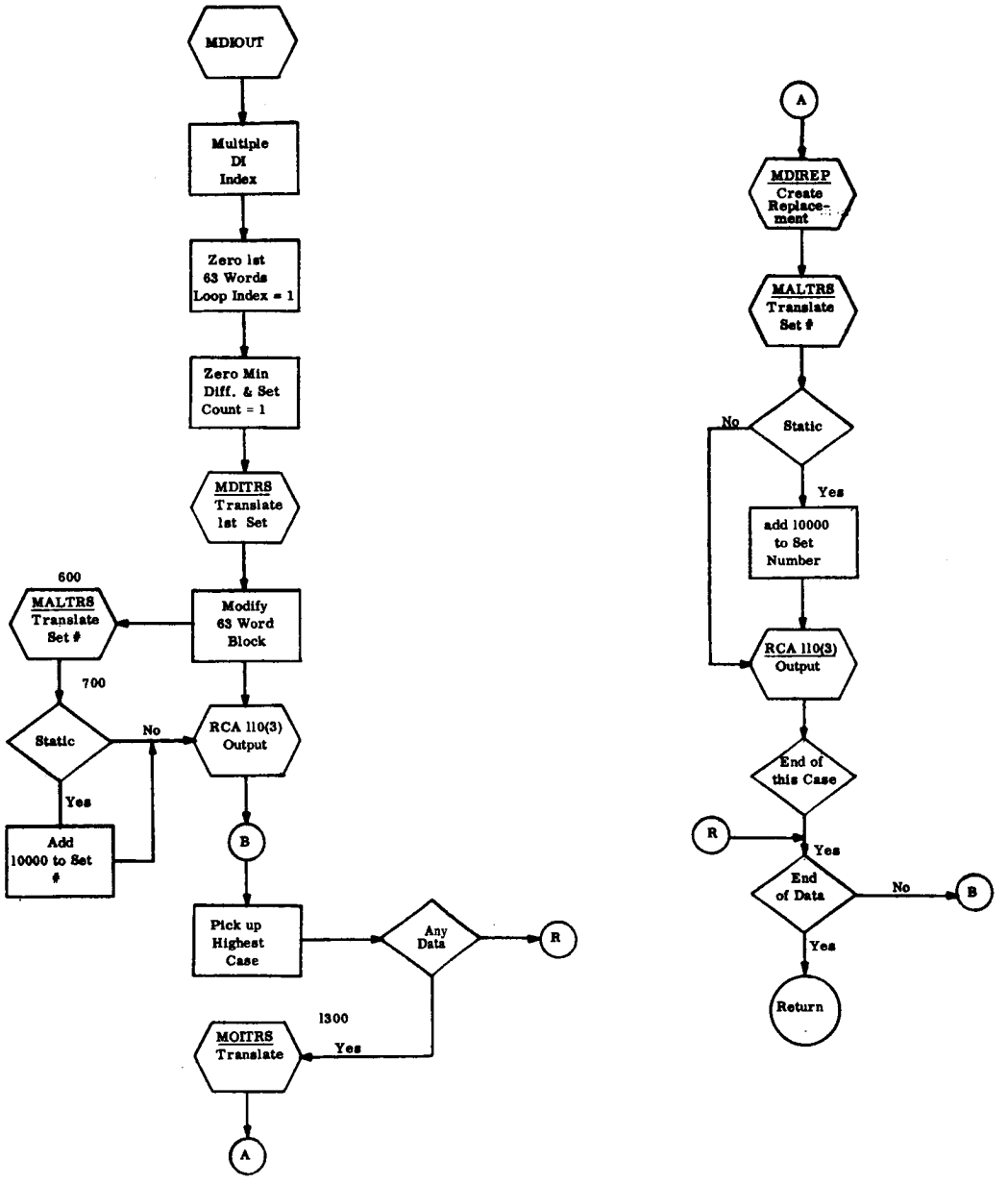


Figure A-16. MDIOUT

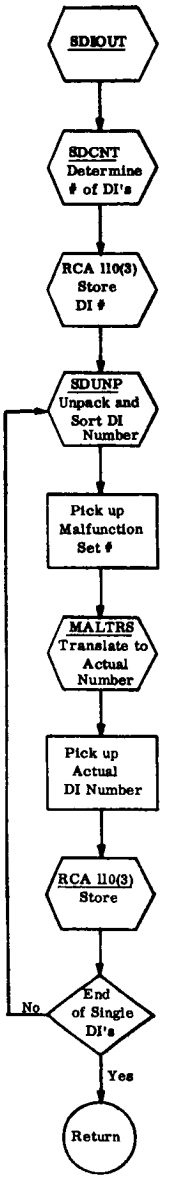


Figure A-17. SDIOUT

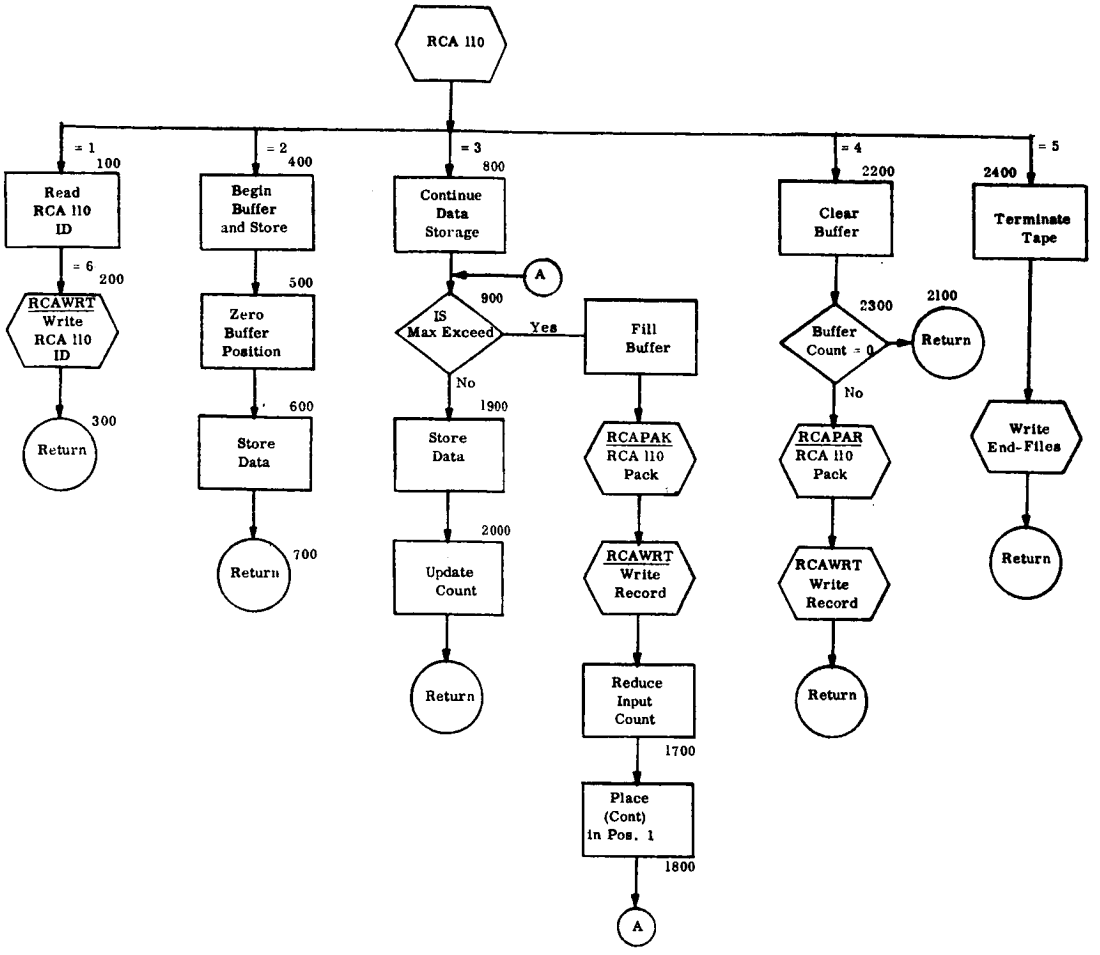


Figure A-18. RCA110

APPENDIX B: DATA MAPS AND STORAGE

I. Data word configurations and referencing mechanisms.

Data from the AMA and DTC-AMA program are condensed into several types of packed format to facilitate data correlation, search, and reordering techniques necessary to produce RCAll0 search tapes.

The major storage area for correlation of data is the labeled common /AMADAT/ which has been assigned 17000 decimal locations. It has two configurations.

/AMADAT/ - Configuration 1: Used during processing for entire test procedure to develop search keys and a unique malfunction set list. The data block has three separate sections.

Section 1: Begins at AMADAT (1). This is an n-entry table having 3 elements/entry where n is equal to the number of active DI's in the test procedure. The configuration of the 3 word entries are (refer to Figure B-1).

Word 1: DI binary number - malfunction set reference.

Word 2: DI bit and position configuration.

Word 3: Chaining control for multiple DI failure effect storage.

Since the DI's are assigned the first index codes by the preceding programs, information is located in this table by implied position based on the three word size).

The decrements of words 1 and words 2 of the entries remain constant for entire run after initial setup. Words 3 and the addresses of words 1 are changed during processing of each substep.

Section 2: Begins after the last entry in the above single DI table. This section is reconstructed for each substep. Access is gained to it through the word 3 items in each of the above entries. It gives the bit-word configuration for the multiple DI failure effects with the correlated malfunction set information. Its length is variable for each substep and as each new multiple DI effect is encountered it is stored in the next available storage locations (see Figure B-2).

Section 3: The creation of this section begins with the first substep in the test procedure and is continually built up during the entire processing. This is the malfunction set section. The first entry in this section is at cell 17000 of AMADAT and storage proceeds backward. Word 1 of each entry is a control word. Word 2-N are the packed forms of the malfunction sets.

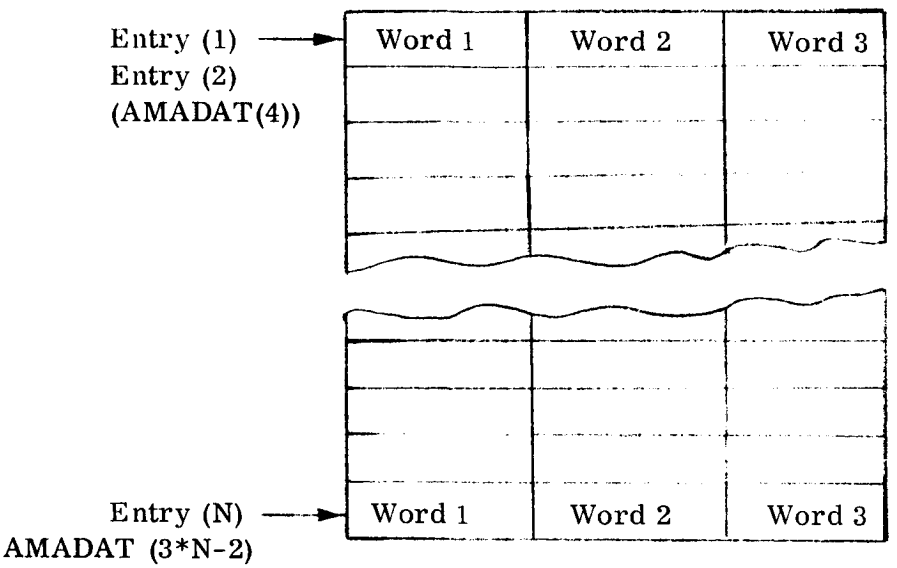
When reference is made to a malfunction set in sections 1 and 2, the reference is made to the address in the table. The control word actually contains the malfunction set number that appears on tape and hard copy. For example, a DI stored at entry 4 of section 1 has malfunction set 16554. This means that AMADAT (16554) contains in its decrement the malfunction set which is actually associated with the DI. (See Figure B-3).

/AMADAT/ - Configuration 2: Set up and used at end of processing of AMA data for complete test procedure.

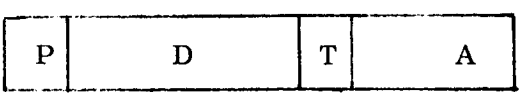
Purpose is to translate the malfunction sets from the internal index codes to the external model names. Core consists of two sections and is set up by the reading of the index code - name translation tape.

Section 1 begins at AMADAT (1) and reference is made by implied position of index code. The contents of each of the cells contains a reference address into the section indicating the beginning of the external name.

The address of the preceding word gives the address of the preceding name. The subtraction of the two gives the number of words in the name. A negative word in the section indicates that the variable was not a failure candidate; therefore its name is not stored, and any reference to it is not translated.



Word 1: Configuration



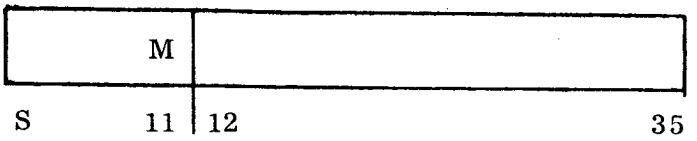
A - Malfunction set number associated with the single effect failure on DI indicated in Decrement

D - Binary number for DI, directly related to model name, ie., if DI108, Decrement contains binary representation of 108

P - Only sign position is used; if negative, the DI has not been processed for the current substep

Figure B-1. /AMADAT/ - single DI configuration.

Word 2:



The word has two sections.

S-11: Indicates that the bit for this DI in its 63 word profile, is contained in the Mth word of the profile.

12-35: Each bit indicates if the DI is turned on in this section. Storage is based on the fact that word 1 contains DI's 23-0 in that order, word 2 contains DI's 41-24 etc.

Word 3: When multiple effect DI failures occur, the effects of the same level can be identified thru use of word 3. The i^{th} entry in the above table gives the storage for i. DI's multiple effect.

A - Gives the address (relative to the beginning of AMADAT of the first word of the first multiple effect information of the i^{th} level).

D - Decrement gives the last item of the i^{th} level.

Figure B-1. /AMADAT/ - single DI configuration (Continued).

Each item is an entry for a unique multiple failure effect for a substep. It may consist of 2 to 64 words. Each item has a lead control word.

S 5	6 20	21 35

S - 5: Number of words in profile to express all the DI's in this effect.

6 - 20: Malfunction set associated with this failure effect.

21 - 35: The address of the next storage location containing an effect failure of the same level. (If 0, this is the last).

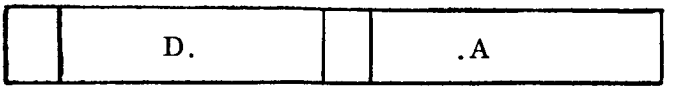
The succeeding words look exactly like the word 2 items of the first entry, except for minimal storage, there may be more than one bit turned on.

Example: Multiple effect failure which affects 4 DI's whose external numbers are 1, 24, 47, 62, has the configuration:

3	malfunction set	next 4-level failure
1	0 ———	——— 010
2	10 ———	——— 01
3	0 ——	01000000000000000000

Figure B-2. /AMADAT/ multiple DI failure effect configuration.

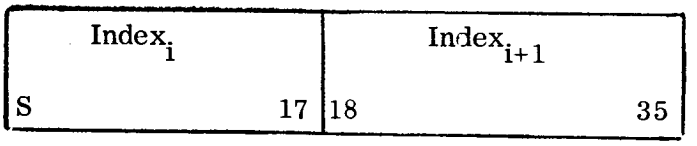
Control Word format:



A - # of words associated with the malfunction set.

D - The external malfunction set number.

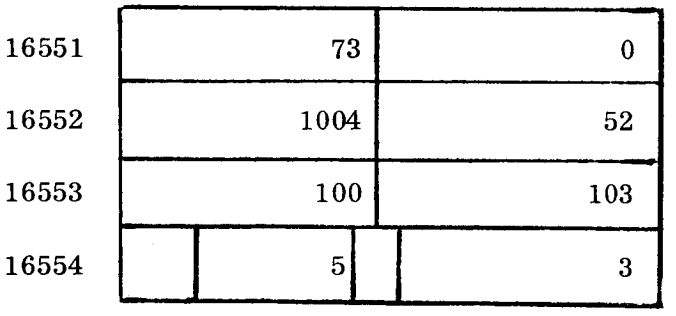
Malfunction Set Configuration Word.



Each index code is stored as an 18 bit binary number. The second index code for a set may be zero.

Example: The malfunction stored at location 16554 is external set 5 and has 5 components as possible failures, whose index codes are 100, 103, 1004, 52 and 73

Location:



The next unique malfunction set encountered is stored starting in cell 16550.

Figure B-3. Malfunction sets storage.

II. Labeled Common

Major communication between routines, tape, and /AMADAT/ is accomplished through a set of labeled commons with definite functions and usages.

1. I/Ø Control - Data passing between tape and core goes into/out of two common sets.

/DNSDAT/AMATYP,DNSCNT,DNSDAT(250)

Data generated by DTC and AMA programs is placed in this data region.

/IRCA/IRCA(255) data to be transmitted to any RCA110 tape is first stored and packed in this common.

2. Data Control Commons for AMADAT

/AMACTR/NSDI - Number of active DI's in procedure.

NMDI - Number of multiple DI's for substep.

NMPØS- Current position (+1) for storage of multiple DI failure effect storage.

/MALMAS/MLSETS - Number of malfunctions in test procedure.

MLPØST - Current position in storage of malfunction sets.

3. Run Control Commons

/MSA/MSA - type run 1, static

2, active

3, merge

/IDDAT/TESTP - test procedure number (BCD)

BLØCK(2) - BLØC and block number (BCD)

STEP - Step (BCD)

SBSTEP - Substep (BCD)

/PANDIS/PARDIS(63) - Active DI Codes

NØNPAR(63) - Static DI codes

DIØNØF(63) - On/off states for active DI's

4. Communication Commons

/FAILDI/NDI - Number of DI's referenced by EFFECT record.

NCØD - Reference position of single DI effect or number of words involved in multiple DI effect.

IDUP - Negative indicates the effect has been encountered previously in processing (signal to extend malfunction set data).

DICØDE - Multiple DI effect configuration for current EFFECT record, or replacement search key.

/MDICON/MDIWRD - Number of words in multiple DI configuration storage.

MDIMAL - Malfunction set address associated with effect.

MDIADD - Address of multiple effect of same level.

MDIPØS - Position in storage of configuration.

/MALSET/MALCNT- Number of words/components associated with malfunction set currently being processed.

MALSET - Address of set or actual external number.

MALPØS - Address of set.

MALCTT - Malfunction set control word.

/NAME/DTCTYP - type of data from DTC index-variable generated tape.

If = 1, beginning of names.

= 2, end-of names.

= 3, name.

NAMLEN - number of words in name

NAME(8) - name in either 7090 form or RCA110 form.

INDEX - Internal index code and failure candidate information.

/DICØN/DIBIN - corresponding binary number for DI

DICØN - total configuration positioning for DI

DIBIT - Bit configuration

DIWRD - Word positioning in 63 word profile.

DIMAL - Malfunction set address associated with single DI effect.

/REPLAC/REPLAC (129) search key replacement words for multiple effect failures.

I. Index-variable translation tape is outlined with DNS-AMA-DTC Program (3843A).

II. AMA data tape is outlined with AMA program (3998).

III. RCA110-AMA tape - RCA 110 format (4 characters per word).

1. Record 1 - RCA110 Identification (16 words)

2. Record 2 - Word 1 TEST

Word 2 Test Number

Word 3 BLØC

Word 4 First block in tape

Words 5-67 - 63 word profile for active DI's

Words 68-130 - 63 word profile for static DI's

3. Data beginning with record 3 is separated into logical blocks for sub-step; two types of logical records;

Record 1: Identification and single DI information.

Word 1: BLØC

Word 2: Block number

Word 3: Step number

Word 4: Sub-step number

Word 5: Number of single DI effects in this record.

Words 6-N in groups of two for as many groups as indicated by word 5.

Word 1 of group - binary representation of DI

Word 2 malfunction set number

Record 2: Multiple effect information

Word 1: MULT

Word 2: Number of multiple effects in record.

Words 3 - 65: 63 word profile for first multiple effect.

Word 66 : Associated malfunction sets for above.

Word 67 : Begins replacement word keys. Each replacement key has the following format:

Word 1: Number of replacements to make.

Words 2-N: Groups of two words/replacement

First word is position of word to replace.

Second word is the replacement word.

Word N+1: Malfunction set associated with 63-word profile after replacement. Sequence is repeated for as many replacements as indicated by word 2 (-1).

If either of the two logical records require more than one physical record, the first word of the physical record is CØNT and data continues in Word 2.

4. The tape terminates with either

Word 1 : ENDB

Word 2 : Block number (last in tape)

Words 3-6: Fill in

which indicates that another physical tape follows for the test procedure, or

Word 1 : ENDT

Word 2 : Test Number

Words 3-6: Fill in

which indicates end of test procedure. Tapes terminate with double end-of-file marks.

IV. Malfunction Set Tape.

1. Record 1 : RCA110 Identification (16 words)
2. Record 2 : Word 1 : TEST
   Word 2 : Test Number
   Words 3-6: Fill in
3. Records 3-N : AMA static data in the same form as the two logical records of III-3 with the four words of identification.
4. Record N+1 : Words 1-2: MALFUNCTIONS
   Words 3-6: Fill in
5. Record N+1 to M: Malfunction set data of the form:
   One logical record/malfunction set
   Word 1 : Set Number
   Word 2 : Number of components
   Words 3-K: For each component, 6 words (blank words if name does not fill 6 words)
6. Tape terminates with a double end-of-file.

V. Static data tape is generated by the program under the static option and contains information of both tapes III and IV. Tape is only used with the 7090 program.

Record 1 : RCA110 identification.
Record 2 : 63 7090 words for static profile.
Records 3-N : AMA in form described in III-3.
Record N+1 : Same as IV-4.
Records N+2-M: Same as IV-5.